Tratamento do Volume de Dados Armazenados em Ambiente de Aprendizado Baseado em Instâncias

Luis André Claudiano

Janeiro / 2018

Dissertação de Mestrado em Ciência da Computação

Tratamento do Volume de Dados Armazenados em Ambiente de Aprendizado Baseado em Instâncias

Esse documento corresponde à dissertação apresentada à Banca Examinadora no curso de Mestrado em Ciência da Computação da Faculdade Campo Limpo Paulista.

Campo Limpo Paulista, 16 de Janeiro de 2018.

Luis André Claudiano

Profa. Dra. Maria do Carmo Nicoletti

Orientadora

FICHA CATALOGRÁFICA

Dados Internacionais de Catalogação na Publicação (CIP)

Câmara Brasileira do Livro, São Paulo, Brasil.

Claudiano, Luis André

Tratamento do volume de dados armazenados em ambiente de aprendizado baseado em instâncias / Luis André Claudiano. Campo Limpo Paulista, SP: FACCAMP, 2018.

Orientadora: Profª. Drª. Maria do Carmo Nicoletti
Dissertação (Programa de Mestrado em Ciência da Computação) – Faculdade Campo Limpo Paulista – FACCAMP.

1. Aprendizado indutivo de máquina. 2. Aprendizado baseado em instâncias. 3. Redução do volume de instâncias. I. Nicoletti, Maria do Carmo. II. Campo Limpo Paulista. III. Título.

CDD-005.75

Resumo. *Algoritmos que implementam ABI têm que lidar com o problema de decidir quais instâncias de dados (e suas descrições) armazenar para uso durante a fase de generalização. O armazenamento de muitas instâncias requer uma memória de tamanho considerável, fato que pode contribuir para tornar mais lenta a execução do programa que implementa tal algoritmo. Assim sendo, estratégias devem ser consideradas que permitam a redução em exigências de armazenamento, sem que isso provoque redução da performance de tais programas. O trabalho de pesquisa apresenta a investigação empiricamente de algoritmos que implementam técnicas de redução de armazenamento em algoritmos que são derivados do algoritmo NN. Os resultados obtidos no experimento realizado são apresentados e discutidos, considerando a redução de armazenamento e a precisão da classificação.*

Abstract*: Algorithms that implement IBL frequently have to deal with the problem of deciding which data instances (and their description) to store for use during the generalization phase. Storing too many instances requires large memory, which can contribute to slowing down the execution speed. So strategies that allow a reduction in the storage requirements, without reducing the performance of programs implementing such algorithms, should be considered. The research work presents the empirical investigation of algorithms that implement storage reduction techniques in algorithms that are derived from the NN algorithm. The results obtained in the realized experiment are presented and discussed, considering the reduction of storage and the precision of the classification.*

Agradecimentos

Agradeço primeiramente a Deus pelos dons concebidos ao longo dessa caminhada.

À minha família, aos meus pais João Expedito Claudiano (*In memoriam*) e Vilma de Fatima Almeida Claudiano, a minha esposa Patricia Maris dos Santos Claudiano, pela paciência e compreensão em todos os momentos dessa caminhada, a minha “filha de quatro patas” Amora por estar sempre acompanhando cada passo dessa dissertação.

À Professora e Orientadora Dra. Maria do Carmo Nicoletti pela orientação, dedicação, ensinamentos (não apenas sobre o conteúdo da dissertação e sim de vida e profissionalismo), incentivo e principalmente pela paciência.

Ao Professor Dr. Sérgio Donizetti Zorzo pelas contribuições e prontidão com o trabalho examinado.

À instituição FACCAMP, ao coordenador do programa de mestrado em Ciência da Computação Prof. Dr. Osvaldo Luiz de Oliveira, a todos professores e funcionários.

A todos aqueles que de uma forma ou de outra incentivaram para que a realização deste trabalho fosse possível, principalmente aos amigos, Professores da Universidade Metodista de Piracicaba UNIMEP e aos meus alunos que me ensinam com suas dúvidas.

SUMÁRIO

Lista de Siglas

ABI – Aprendizado Baseado em Instâncias

AIM – Aprendizado Indutivo de Máquina

AM – Aprendizado de Máquina

IA – Inteligência Artificial

DC – Descrição do Conceito

SABI – Sistema para Aprendizado Baseado em Instâncias

Lista de Tabelas

Lista de Figuras

Lista de Algoritmos

Capítulo 1
Introdução

Aprendizado de Máquina (AM) é uma das muitas subáreas da área de pesquisa denominada Inteligência Artificial (IA). O chamado *aprendizado indutivo de máquina* (AIM) é o modelo de AM mais popular e mais bem-sucedido e tem sido implementado utilizando inúmeras técnicas e algoritmos (ver [Mitchell 1997]). Para viabilizar AIM é mandatório que um conjunto de instâncias, que representam os conceitos a serem aprendidos, esteja disponível. Esse conjunto de instâncias é denominado *conjunto de treinamento*. Conjuntos de treinamento geralmente são descritos por vetores de pares atributo−valor_de_atributo e, dependendo da situação, de uma classe associada (que indica qual conceito a instância em questão representa). A classe de cada instância que participa do conjunto de treinamento é, na maioria dos casos, determinada por um especialista humano da área de conhecimento descrita pelos dados.

Dentre os muitos modelos de aprendizado supervisionado (*e.g.*, simbólico, neural, etc.), encontra-se aquele chamado de *aprendizado baseado em instâncias* (ABI) (também conhecido como aprendizado preguiçoso (*lazy learning*)). Como comentado em [Mitchell 1997] algoritmos que implementam ABI, via de regra, simplesmente armazenam as instâncias de treinamento e adiam o processo de 'generalização' até o momento em que uma nova instância, de classe desconhecida, precisa ser classificada. Uma vantagem desse tipo de aprendizado é que, ao invés de generalizar o conceito, considerando todo o conjunto de treinamento disponível, estima o conceito localmente, para cada nova instância a ser classificada. Uma das desvantagens de algoritmos baseados em instâncias é o custo da classificação, quando o conjunto de treinamento é volumoso e envolve, também, um grande número de atributos, uma vez que todo o processamento requerido acontece na fase de classificação, dado que a fase de aprendizado consiste simplesmente no armazenamento do conjunto de treinamento.

Esta dissertação descreve o trabalho de pesquisa realizado em nível de mestrado, que teve como contexto os algoritmos de ABI e, com foco, a investigação de esquemas que buscam tratar/contornar um dos principais inconvenientes no uso de tais algoritmos

i.e., aquele decorrente da necessidade de armazenamento de todas as instâncias de treinamento. Uma das formas de lidar com o alto volume de instâncias armazenadas é por meio das chamadas técnicas de redução. Para tanto, o trabalho se propôs a investigar empiricamente algoritmos que implementam técnicas de redução, particularmente os seis algoritmos propostos em [Wilson & Martinez 2000]. A investigação tem como objetivo verificar a real contribuição de tais estratégias, quando agregadas a um conjunto de algoritmos que implementam ABI.

Além desse capítulo introdutório, o documento é composto por mais sete capítulos, cujas respectivas identificações e conteúdos são: Capítulo 2 contextualiza o foco da pesquisa *i.e.*, ABI, como parte integrante da área de Aprendizado Indutivo de Máquina e, para tanto, apresenta alguns conceitos básicos de AIM necessários ao entendimento, bem como uma caracterização mais detalhada dos algoritmos de ABI. Este capítulo também apresenta um panorama dos principais representantes da família de algoritmos ABI. O Capítulo 3 considera algumas das estratégias encontradas na literatura para tratar o problema de redução de volume de dados em ABIs.

No Capitulo 4 é feita a abordagem do algoritmo IB2, que busca minimizar o volume de instâncias armazenadas, para demonstrar os processos envolvidos são apresentados exemplos de sua utilização e a reconstituição do experimento realizado em [Aha *et al.* 1991], discutindo os conceitos envolvidos.

O foco do Capitulo 5 são os algoritmos CNN, RNN e $Method_{2,}$ que são baseados no algoritmo NN (*Nearest Neighbour*), e utilizam estratégias que buscam minimizar o volume de instâncias a serem armazenadas.

O Capítulo 6 descreve o sistema computacional SABI (*Sistema para Aprendizado Baseado em Instância*s), que foi implementado com os algoritmos: NN, IB1, IB2, CNN, RNN e $Method_2$, disponibilizando um ambiente para a realização de experimentos para os algoritmos ABI.

Para avaliar as estratégias de redução de volume de instâncias armazenadas, o Capitulo 7 descreve o experimento realizado nessa dissertação e analisa os resultados, apontando um comparativo do impacto do armazenamento *versus* precisão.

As considerações e conclusões sobre o trabalho realizado, bem como sugestões de pontos a serem investigados como continuidade a pesquisa realizada descrita nesta dissertação são apresentadas no Capitulo 8.

Capítulo 2
Aprendizado Indutivo de Máquina e Algoritmos de Aprendizado Baseado em Instâncias (ABI) – Caracterização, Principais Conceitos e Algoritmos

2.1 Considerações Iniciais

Esse capítulo está organizado em quatro seções. A Seção 2.2 tem foco na apresentação dos principais fundamentos associados à área de pesquisa de Aprendizado Indutivo de Máquina (AIM), uma subárea da área de Inteligência Artificial. A Seção 2.3 discute as principais características de um conjunto de algoritmos de AIM, caracterizados como de aprendizado baseado em instâncias (ABI). Particularmente dois deles, o NN (*Nearest Neighbour*) e o IB1 (*Instance Based 1*), que são referências a muitos outros que se seguiram, são apresentados e discutidos nas seções 2.4 e 2.5, respectivamente. Ambos os algoritmos são semelhantes, a menos de pequenos detalhes e serão abordados na segunda parte do Capítulo 4, em combinação com os algoritmos de redução, que são o foco desse trabalho.

2.2 Aprendizado Indutivo de Máquina

Aprendizado Indutivo de Máquina (AIM) é o modelo de AM mais popular e bem sucedido; tem sido implementado utilizando inúmeras técnicas e algoritmos (como apresentado em [Mitchell 1997]). Como comentado anteriormente, algoritmos caracterizados como de AIM recebem como *input* um conjunto de instâncias de dados, que representam os conceitos a serem aprendidos, denominado *conjunto de treinamento*.

Conjuntos de treinamento geralmente são descritos por vetores de pares *atributo–valor_de_atributo* e, dependendo da situação, de uma classe associada (que indica qual conceito a instância em questão representa). A classe de cada instância que participa do conjunto de treinamento é, na maioria dos casos, determinada por um especialista humano da área de conhecimento descrita pelos dados.

O fato de a classe participar da descrição da instância e do algoritmo de aprendizado fazer uso dessa informação caracteriza tal algoritmo como *algoritmo de aprendizado supervisionado*. Em muitas situações do mundo real, entretanto, a classe à qual cada instância pertence é desconhecida e/ou não existe um especialista humano que, com base na descrição dos valores de atributos, seja capaz de determinar a classe associada àquela a instância. Algoritmos de AM que lidam com conjuntos de instâncias que não têm uma classe associada são conhecidos como algoritmos *de aprendizado não-supervisionado* (dentre os inúmeros existentes o mais popular é conhecido como agrupamento (*clustering*)).

A grande maioria dos algoritmos de AM supervisionado são caracterizados como classificadores e operam da seguinte forma: dado um conjunto de treinamento contendo descrições de N instâncias do conceito a ser aprendido, cada uma delas caracterizada como um conjunto de M atributos, induzem uma generalização do conjunto de treinamento, cuja representação varia de algoritmo para algoritmo. Esse processo é usualmente referenciado como *fase de aprendizado*. Se o algoritmo utilizado for um algoritmo de treinamento de uma rede neural, a representação do conjunto de treinamento é a própria rede, por meio de sua topologia (número de camadas e de neurônios/camada), peso das conexões e outras possíveis informações. Vários algoritmos de AM induzem a generalização do conjunto de treinamento como um conjunto de regras (ver, por exemplo, o CN2 [Clark & Niblett 1989]), outros, representam a generalização obtida como uma árvore de decisão (como é o caso do C4.5 [Quinlan 1993]).

Uma vez induzida a expressão do conceito, seja ela expressa em qualquer uma das representações usualmente utilizadas, tal expressão pode ser, então, utilizada para a classificação de novas instâncias de dados com classe desconhecida, em um processo identificado como *fase de classificação*.

Particularmente, um algoritmo de ABI, ao receber como *input* um conjunto de treinamento, não o generaliza – apenas o armazena. O processo de generalização (*i.e.*, o aprendizado em si) é feito no momento da classificação de uma nova instância, de classe desconhecida, como é descrito detalhadamente na Seção 2.3 a seguir.

2.3 Algoritmos Baseados em Instâncias – Caracterização e Conceitos Associados

Dentre os muitos modelos de aprendizado supervisionado (e.g., simbólico, neural, etc.), encontra-se aquele chamado de baseado em instâncias (e também conhecido como aprendizado preguiçoso). Como comentado em [Mitchell 1997], algoritmos que implementam aprendizado baseado em instâncias simplesmente armazenam as instâncias de treinamento e adiam o processo de 'generalização' até o momento em que uma nova instância, de classe desconhecida, precisa ser classificada.

Para classificar a nova instância os algoritmos ABI fazem uma varredura do conjunto de instâncias armazenadas, buscando identificar aquela que seja mais 'similar' à nova instância. Via de regra tais algoritmos usam como critério de 'similaridade' o conceito de proximidade, implementado como a distância euclidiana. Ou seja, o algoritmo busca, entre as instâncias armazenadas, aquela que está mais próxima da nova instância; a classe da instância identificada como 'mais próxima' é, então, atribuída à nova instância que, até então, tinha classe desconhecida. Uma *descrição do conceito baseada em instâncias* inclui o conjunto de instâncias armazenadas e, possivelmente, alguma informação relativa ao desempenho de tais instâncias, durante processos de classificação feitos anteriormente.

Uma vantagem do aprendizado via algoritmos ABI é que, ao invés de generalizar o conceito, considerando todo o conjunto de treinamento disponível, estima o conceito localmente, para cada nova instância de dado a ser classificada. Uma das desvantagens de algoritmos baseados em instâncias é o custo da classificação, quando o conjunto de treinamento é volumoso, uma vez que todo o processamento envolvido acontece na fase de classificação, já que a fase de aprendizado de algoritmos ABI consiste simplesmente no armazenamento do conjunto de treinamento.

Como apontado em [Santos & Nicoletti 1997], entre algumas das dificuldades de algoritmos ABI estão o aumento do tempo de classificação, à medida que muitas instâncias de treinamento vão sendo adicionadas à memória e, também, a possibilidade de tais algoritmos excederem a capacidade da memória disponível, devido ao grande número de instâncias armazenadas.

2.4 O Algoritmo *Nearest Neighbour* (NN)

Algoritmos que implementam o ABI são, via de regra, inspirados no algoritmo *Nearest Neighbour* (NN) [Cover & Hart 1967] ou em alguma de suas variantes [Hart 1968] [Gates 1972]. Esta seção apresenta e discute o NN e mostra um exemplo de seu uso em uma situação trivial, com o intuito de mostrar o seu funcionamento.

2.4.1 Considerações sobre o NN

O NN (*Nearest Neighbor-Vizinho Mais Próximo)*, cuja descrição em pseudocódigo está em Algoritmo 2.1, é o algoritmo mais básico de ABI e, como comentado em [Gates 1972], sua popularidade se deve "... à sua simplicidade conceitual, a qual leva a uma implementação direta, embora não necessariamente, mais eficiente." Tal algoritmo assume que todas as instâncias armazenadas correspondem a pontos (*i.e.*, instâncias de dados) no espaço N-dimensional R^N e, então, uma função de distância é usada para determinar qual instância de dado, dentre aquelas armazenadas, é a mais próxima de uma dada nova instância (a ser classificada). Uma vez determinada a instância mais próxima, sua classe é atribuída à nova instância considerada. Via de regra a determinação do(s) vizinho(s) mais próximo(s) é feita por meio do cálculo da distância; a distância euclidiana é a mais popular entre as distâncias utilizadas para esse fim.

Considere:
- espaço M-dimensional de atributos
- S classes, numeradas 1,2,3,...,S
- N instâncias de treinamento, cada uma representada como um par (x_i,θ_i), $1 \le i \le N$, tal que:

(a) x_i é uma instância representada por um vetor de M valores de atributos:

$$x_i = (x_{i_1}, x_{i_2}, ..., x_{i_M})$$

(b) $\theta_i \in \{1,2,...,S\}$, $1 \le i \le N$, denota a classe correta da instância x_i

Seja $TNN = \{(x_1, \theta_1), (x_2, \theta_2), ..., (x_N, \theta_N)\}$ o conjunto de treinamento do NN. Dada uma instância desconhecida x, a regra de decisão do algoritmo decide que x está na classe θ_j se

$$d(x, x_j) \le d(x,x_i) \qquad 1 \le i \le N$$

em que d é alguma métrica M-dimensional de distância.

Algoritmo 2.1 Algoritmo NN como originalmente proposto em [Cover & Hart 1967].

A regra de decisão que estabelece que a classe de uma nova instância de dado x é θ_j (*i.e.*, $d(x, x_j) \le d(x,x_i)$, $1 \le i \le N$), parte do Algoritmo 2.1, é mais apropriadamente chamada de

regra 1-NN, uma vez calcula a distância da nova instância a cada uma das instâncias do conjunto de treinamento e, para associar a classe à nova instância, considera a classe de apenas *uma* instância; aquela que está mais próxima da nova instância.

Considere que uma instância arbitrária x_i seja descrita pelo vetor M-dimensional de valores de atributos notado por $<x_{i_1}, x_{i_2}, \ldots, x_{i_M}>$, em que x_{i_r} representa o valor do r-ésimo atributo da instância x_i, $1 \leq r \leq M$). A distância euclidiana entre duas instâncias x_i e x_j é mostrada na Eq. (2.1).

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{M} (x_{i_r} - x_{j_r})^2} \qquad (2.1)$$

O processo de classificação de uma nova instância pode ser estendido considerando um número maior de vizinhos mais próximos (da instância a ser classificada), o que dá origem à versão do algoritmo conhecida como *k vizinhos mais próximos* (k-NN), cujo algoritmo, para o aprendizado de funções com valores discretos *i.e.*, f: $R^M \rightarrow V$, em que $V=\{v_1,\ldots,v_S\}$ é apresentado em Algoritmo 2.2 (extraído de [Mitchell 1997]).

Algoritmo de treinamento:

para cada exemplo de treinamento (x,f(x)), inclua o exemplo na lista *exemplos_treinamento*

Algoritmo de classificação(k,x_q)

% k: número de vizinhos mais próximos a ser considerado
% x_q: instância a ser classificada,

- sejam $x_1,\ldots,x_k$ as k instâncias da lista *exemplos_treinamento* mais próximas de x_q
- retorne

$$\hat{f}(x_q) \leftarrow \operatorname{argmax}_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$$

em que $\delta(a,b) = 1$ se a = b, caso contrário, $\delta(a,b)=0$

Algoritmo 2.2 Algoritmo k-NN para aproximação de uma função com valores discretos, $f:R^M \rightarrow V$.

Considerando o algoritmo descrito no Algoritmo 2.2, seja *exemplos_treinamento* o conjunto $\{(x_1, v_1), (x_2,v_1), (x_3,v_2), (x_4,v_3), (x_5,v_1)\}$ ou seja, um conjunto com cinco instâncias x_1, x_2, x_3, x_4, x_5, representando 3 classes distintas *i.e.*, v_1, v_2 e v_3 (note que no exemplo, a dimensionalidade de cada instância está sendo abstraída) e que o valor considerado para k seja 3.

Na descrição em Algoritmo 2.2, a função f associa a cada instância sua

respectiva classe ou seja, $f(x_1) = v_1$, $f(x_2) = v_1$, $f(x_3) = v_2$, $f(x_4) = v_3$, $f(x_5) = v_1$. Suponha que x_6 seja uma instância a ser classificada e que x_1, x_2 e x_4 sejam as k=3 instâncias mais próximas de x_6. Note que x_1 e x_2 têm a mesma classe, que é v_1, e que a classe de x_4 é v_3. De acordo com o Algoritmo 2.1, uma soma deve ser realizada, para cada uma das classes associadas às k=3 instâncias existentes, ou seja, para as classes v_1 e v_3.

Para a classe v_1 soma-se, para cada uma das instâncias próximas de x_6, os valores $\delta(v_1,f(x_1)) + \delta(v_1,f(x_2)) + \delta(v_1,f(x_4)) = \delta(v_1,v_1) + \delta(v_1,v_1) + \delta(v_1, v_3) = 1+1+0=2$.

Para a classe v_3 soma-se, para cada uma das instâncias próximas de x_6, os valores $\delta(v_3,f(x_1)) + \delta(v_3,f(x_2)) + \delta(v_3,f(x_4)) = \delta(v_3,v_1) + \delta(v_3,v_1) + \delta(v_3, v_3) = 0+0+1=1$.

Dentre as duas classes $v \in \{v_1, v_3\}$, a que teve o maior valor de soma foi a classe v_1 e, portanto, tal classe é atribuída à instância x_6. A Figura 2.1 ilustra o exemplo.

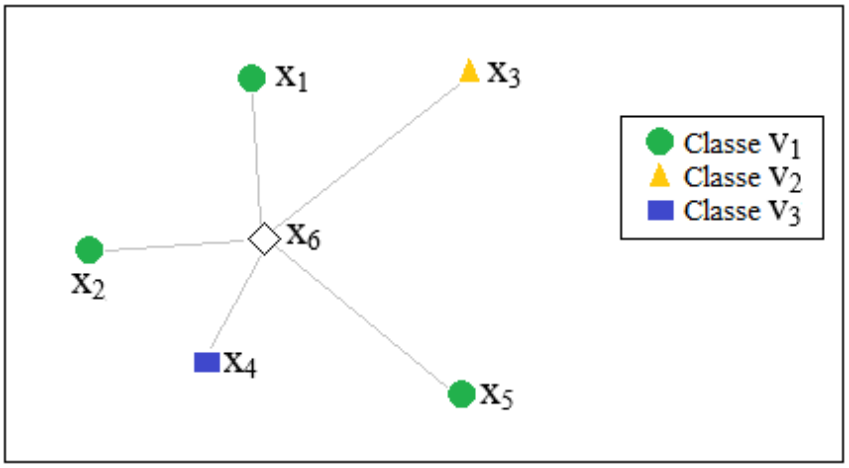


Figura 2.1 Classificação da instância x_6, considerando o 3-NN. Como os três vizinhos mais próximos de x_6 são x_1, x_2 e x_4, e a classe majoritária entre essas três instâncias é v_1, x_6 será classificado como da classe v_1.

O Algoritmo 2.1 pode ser facilmente adaptado para a aproximação de funções com valores contínuos. Para fazer isso o algoritmo deve calcular o valor médio dos k exemplos de treinamento mais próximos, ao invés de calcular o seu valor mais comum. Mais precisamente, para aproximar uma função contínua $f: R^M \rightarrow R$, a fórmula final do Algoritmo 2.1 deve ser substituída por Eq. (2.2).

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k} \qquad (2.2)$$

Quando do cálculo da distância entre instâncias, o k-NN utiliza todos os atributos que descrevem a instância. Este fato contrasta, de certa forma, com outros métodos de

aprendizado automático como, por exemplo, o aprendizado de regras de decisão (ou árvores de decisão). Algoritmos que constroem árvores de decisão utilizam medidas (via de regra, a entropia) para identificar aqueles atributos que têm maior relevância para caracterizar os conceitos sendo aprendidos e, desta maneira, nem sempre o conjunto inteiro de atributos que descreve o conjunto de instâncias de treinamento, comparece na expressão induzida do conceito. Essas medidas tendem a descartar atributos que são irrelevantes (para a caracterização do conceito).

2.4.2 Um Exemplo de Uso do NN

No que segue é apresentado um pequeno exemplo do uso do algoritmo NN, em que o conjunto de treinamento fornecido está descrito na Tabela 2.1, contendo 20 instâncias de dados, sendo 9 delas de classe 1 e 11 de classe 2. O NN, como comentado anteriormente, durante a fase de treinamento apenas armazena as instâncias do conjunto de treinamento. A expressão do conjunto nada mais é do que o próprio conjunto de treinamento mostrado na Tabela 2.1, que pode ser visualizado na Figura 2.2.

Tabela 2.1 Conjunto de treinamento armazenado pelo algoritmo NN contendo 20 instâncias de dados, sendo 9 de classe 1 e 11 de classe 2.

#ID	X	Y	Classe	#ID	X	Y	Classe
I_1	1,1	3,2	1	I_{11}	5,3	7,5	2
I_2	5,3	7,1	2	I_{12}	5,7	7,8	2
I_3	1,8	3,9	1	I_{13}	1,3	3,3	1
I_4	5,5	7,8	2	I_{14}	1,3	3,7	1
I_5	5,5	7,2	2	I_{15}	5,7	7,2	2
I_6	2,0	4,0	1	I_{16}	5,2	7,3	2
I_7	1,5	3,5	1	I_{17}	1,3	3,1	1
I_8	1,6	3,1	1	I_{18}	1,3	6,1	1
I_9	5,1	5,9	2	I_{19}	6,7	7,9	2
I_{10}	5,8	7,1	2	I_{20}	5,4	7,4	2

Quando da classificação de uma nova instância, o algoritmo calcula a distância da nova instância a todas outras do conjunto de treinamento e a nova instância é classificada com a classe da instância armazenada, que lhe for mais próxima. Na Figura 2.2, as novas instâncias consideradas são: NI_1(2,6 1,3), NI_2(5,2 3,2) e NI_3(3,5 4,5) que serão classificadas com classes: 1, 2 e 1, respectivamente.

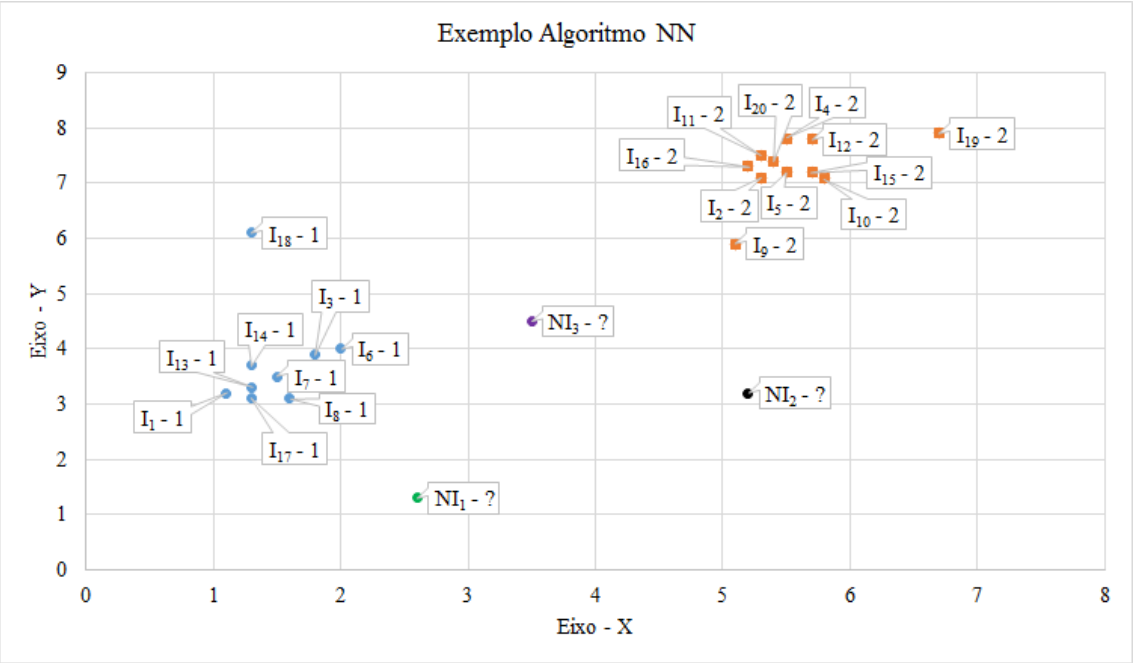


Figura 2.2 Conjunto das 20 instâncias de treinamento (Tabela 2.1) e as novas instâncias de dados, NI_1(2.6,1.3), NI_2(5.2,3.2) e NI_3(3.5,4.5) aguardando a classificação e que, finalmente, serão classificadas como de classe 1, 2 e 1.

2.5 A Família IBL e o Algoritmo IB1

Na literatura podem ser encontradas inúmeras propostas de variantes do NN (e k-NN) (ver [Bhatia & Vandana 2010]), que buscam contornar algum(ns) dos problemas apontados na Seção 2.3. Com esse mesmo objetivo e, também, para melhor caracterizar e tornar algoritmos de ABI mais robustos, Aha e colaboradores conduziram um estudo descrito em [Aha *et al.* 1991, Aha 1992] em que vários algoritmos de ABI foram propostos, sendo o IB1 o primeiro e o mais simples deles; os que o seguem *i.e.*, IB2, IB3, IB4 e IB5, tratam de mecanismos para contornar alguns dos problemas evidenciados no uso de tais algoritmos em domínios reais, tais como tratamento de ruídos nos dados, tratamento de atributos irrelevantes e a introdução de novos atributos.

As três seções que compõem a Seção 2.5 abordam, respectivamente, uma descrição geral de uma família de algoritmos baseados em instâncias conhecida como IBL (*Instance-based Learning)* (Seção 2.5.1), o primeiro algoritmo da família, chamado IB1(Seção 2.5.2) e o uso do IB1 em uma situação trivial (Seção 2.5.3).

2.5.1 A Família *Instance-Based Learning* (IBL)

A família IBL é composta por cinco algoritmos, a saber, IB1, IB2, IB3, IB4 e IB5, que foram propostos por Aha e colaboradores em [Aha *et al.* 1991][Aha 1992], com o objetivo de investigar e descobrir os limites da abordagem de aprendizado que armazena

instâncias de treinamento, sem realizar qualquer tipo de generalização sobre elas. Todas as variantes usam a técnica do 'vizinho mais próximo' para classificar novos exemplos [Santos & Nicoletti 1997]. Os cinco algoritmos da família diferem, principalmente, com relação às estratégias associadas a:

(1) armazenamento das instâncias de treinamento;

(2) avaliação da similaridade entre as instâncias;

(3) número de instâncias usadas quando da classificação de uma nova instância, de classe desconhecida.

Os algoritmos da família IBL procuram contornar alguma das limitações associadas aos algoritmos de IBI que, como apontado em [Breiman *et al.* 1984], tem como inconvenientes:

- são classificadores computacionalmente caros uma vez que armazenam todas as instâncias de treinamento;
- são intolerantes a atributos com ruído;
- são intolerantes a atributos irrelevantes;
- são sensíveis à escolha da função de similaridade;
- não existir maneira natural ou simples de se trabalhar com atributos que tenham valores nominais ou, então, atributos que, por alguma razão, não têm valores associados;
- fornecerem pouca informação útil com relação à estrutura dos dados.

O conjunto de treinamento é fornecido aos algoritmos da família IBL como uma sequência de instâncias, em que cada uma delas é descrita como um vetor de pares *atributo–valor-de-atributo* e uma *classe* associada. No espaço contendo o conjunto de treinamento, o conjunto de todas as instâncias com um mesmo valor (*e.g.* X) associado à *classe*, define a classe X.

No caso dos algoritmos da família IBL, uma descrição de conceito baseada em instâncias inclui o conjunto de instâncias armazenadas e, possivelmente, informação a respeito do desempenho de cada instância em classificações anteriores, representado pelo número de classificações corretas e incorretas feitas. A descrição do conceito pode também incorporar funções de similaridade e de classificação, além do conjunto de instâncias armazenado. O objetivo dessas funções é determinar como o conjunto de instâncias será utilizado para predizer a classe de novas instâncias [Nardin 2003].

Como apontado por Aha e colaboradores em [Aha *et al.*1991], os três componentes que caracterizam todos os algoritmos da família IBL são: o conjunto de treinamento e as funções de similaridade e de classificação, em que:

função de similaridade: responsável pelo cálculo da similaridade entre uma instância do conjunto de treinamento E_i e as instâncias que já participam da descrição do conceito;

função de classificação: responsável por classificar uma instância E_i. Geralmente tal função leva em consideração os registros de desempenho de classificações anteriores das instâncias que descrevem o conceito, bem como o resultado da função de similaridade;

atualizador da descrição do conceito: procedimento responsável por manter os registros de desempenho de classificação e decidir qual instância incluir na descrição do conceito. Como entrada recebe a instância E_i, os resultados da função de classificação e a descrição atual do conceito. É o procedimento responsável pela atualização do conceito.

2.5.2 O Algoritmo IB1

O IB1 é o mais simples algoritmo da família IBL e tem como principais características o fato de processar as instâncias incrementalmente e utilizar uma política de tolerância à ausência de valores de atributos. O pseudocódigo alto nível do IB1 é mostrado no Algoritmo 2.3. No Algoritmo 2.3 a função *similaridade* é definida pela Equação 2.3,

$$similaridade(x_i, x_j) = -\sqrt{\sum_{r=1}^{M} f\left(x_{i_r}, x_{j_r}\right)} \qquad (2.3)$$

em que instâncias são descritas por M atributos. A função f na Eq. (2.3) é definida como $f(x_{i_r},x_{j_r}) = (x_{i_r} - x_{j_r})^2$ para atributos que têm valores numéricos e como $f(x_{i_r},x_{j_r}) = (x_{i_r} \neq x_{j_r})$ para atributos cujos valores são booleanos ou simbólicos. Neste último caso, o valor verdade associado à condição $(x_{i_r} \neq x_{j_r})$ é traduzido como segue: *verdade*: 0 e *falso*: 1 se similaridade for usada e *verdade*: 1 e *falso*: 0 se dissimilaridade for usada. Para um atributo que têm valor ausente em uma das instâncias, assume-se como seu valor, o valor mais diferente do valor presente (na outra instância); caso ambos estejam ausentes, é estabelecido que $f(x_{i_r},x_{j_r}) = 1$.

DC ← ∅
for each x ∈ TR do
1. **for each** y ∈ DC **do**
Sim[y] ← *similaridade*(x,y)
end for
2. y_{max} ← algum y ∈ DC com Sim[y] maximal
3. **if** classe(x) = classe(y_{max})
then classifica ← correto
else classifica ← incorreto
4. DC ← DC ∪ {x}
end for

Algoritmo 2.3 Algoritmo IB1. TR: conjunto de treinamento e DC: descrição do conceito [Aha *et al.* 1991].

Em [Aha *et al.* 1991] é apresentada uma análise teórica do IB1 em que: (1) os autores provam que o IB1 é aplicável a uma vasta classe de conceitos e (2) o limite superior do número de instâncias necessárias para o *aprendizado-pac (Probably Approximately Correct)* de um conceito é polinomial no tamanho da fronteira do conceito.

2.5.3 Um Exemplo de Uso do IB1

De maneira semelhante ao algoritmo NN, o IB1 armazena todas as instâncias do conjunto de treinamento. O IB1, entretanto, armazena o número de classificações corretas e incorretas que cada instância fez, ao longo do processo de treinamento.

No que segue, o mesmo exemplo da Seção 2.4.1, Tabela 2.1, será considerado. A Figura 2.3 mostra os primeiros passos do processo de armazenamento do conjunto de treinamento considerado, realizado pelo IB1, e a Tabela 2.2 mostra o registro de desempenho associado a cada uma das instâncias do conjunto de treinamento.

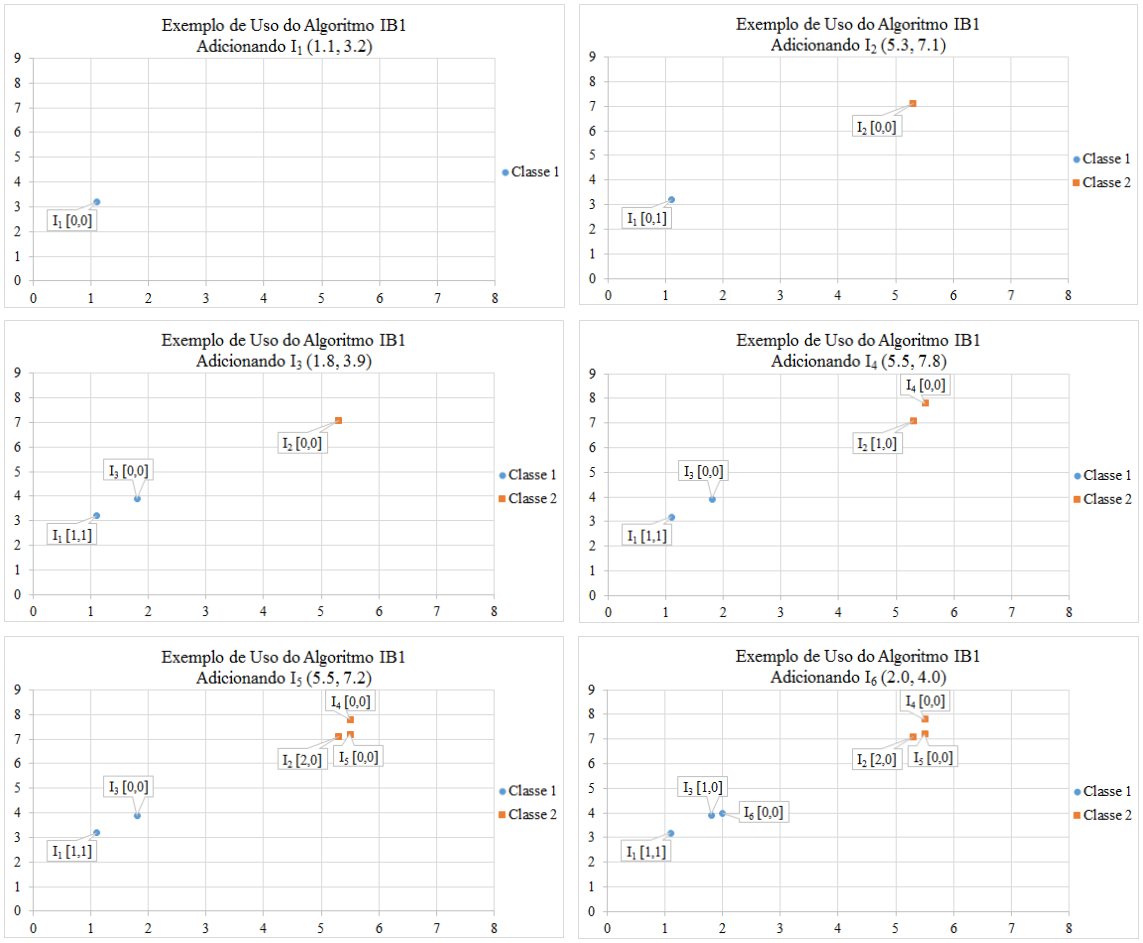


Figura 2.3 Fase de aprendizado do IB1, em que as instâncias de treinamento vão sendo armazenadas e, ao mesmo tempo, usadas na classificações das instâncias que as seguem, como uma estratégia para evidenciar relevância, entre as instâncias. Isso gera, para cada instância, um vetor bidimensional [C I], denominado registro de desempenho, em que C representa o número de classificações corretas e I o número de classificação incorretas que cada instância fez. Ao final, tais números podem identificar instâncias que são mais (ou menos) representativas dos conceitos representados no conjunto de treinamento.

Tabela 2.2 Conjunto de treinamento armazenado pelo algoritmo IB1, mostrando o registro de desempenho associado a cada instância.

#ID	X	Y	[C I]	#ID	X	Y	[C I]
I_1	1,1	3,2	[3,1]	I_{11}	5,3	7,5	[1,0]
I_2	5,3	7,1	[4,0]	I_{12}	5,7	7,8	[1,0]
I_3	1,8	3,9	[1,0]	I_{13}	1,3	3,3	[1,0]
I_4	5,5	7,8	[2,0]	I_{14}	1,3	3,7	[0,0]
I_5	5,5	7,2	[1,0]	I_{15}	5,7	7,2	[0,0]
I_6	2,0	4,0	[1,0]	I_{16}	5,2	7,3	[0,0]
I_7	1,5	3,5	[2,0]	I_{17}	1,3	3,1	[0,0]
I_8	1,6	3,1	[0,0]	I_{18}	1,3	6,1	[0,0]
I_9	5,1	5,9	[0,0]	I_{19}	6,7	7,9	[0,0]
I_{10}	5,8	7,1	[1,0]	I_{20}	5,4	7,4	[0,0]

Note que uma troca na ordem em que os exemplos são entrada para o IB1 não afeta a representação do conceito, uma vez que, para o IB1, o conceito é descrito por todas as instâncias do conjunto de treinamento. No entanto, a troca de ordem pode afetar o registro de desempenho associado a cada instância, como mostrado a seguir. Considere que o mesmo conjunto de treinamento mostrado na Tabela 2.1 seja usado mas que as instâncias em tal conjunto tenham sido 'embaralhadas', como mostra a Tabela 2.3.

A Tabela 2.4 mostra o registro de desempenho criado pelo IB1, quando do uso do conjunto de instâncias mostrado na Tabela 2.3, em que as instâncias são entrada para o algoritmo na sequência mostrada na tabela *i.e.*, I_{18}, I_8, I_7, ..., I_{14}. Comparando os registros de desempenho mostrados nas tabelas 2.2 e 2.4, pode ser evidenciado que uma mesma instância tem dois registros de desempenho diferente, uma vez que a ordem em que ela foi processada pelo IB1 interfere nos valores de tal registro.

Tabela 2.3 Conjunto de treinamento embaralhado, contendo as mesmas 20 instâncias de dados, mostradas na Tabela 2.1, mas em outra ordem.

#ID	X	Y	Classe	#ID	X	Y	Classe
I_{18}	1,3	6,1	1	I_{15}	5,7	7,2	2
I_8	1,6	3,1	1	I_{20}	5,4	7,4	2
I_3	1,8	3,9	1	I_5	5,5	7,2	2
I_7	1,5	3,5	1	I_{13}	1,3	3,3	1
I_1	1,1	3,2	1	I_{17}	1,3	3,1	1
I_{11}	5,3	7,5	2	I_4	5,5	7,8	2
I_{12}	5,7	7,8	2	I_{10}	5,8	7,1	2
I_{16}	5,2	7,3	2	I_{19}	6,7	7,9	2
I_2	5,3	7,1	2	I_9	5,1	5,9	2
I_6	2,0	4,0	1	I_{14}	1,3	3,7	1

Tabela 2.4 Registros de desempenho de instâncias criados pelo IB1, do conjunto embaralhado, mostrado na Tabela 2.3.

#ID	X	Y	Classe	[C I]	#ID	X	Y	Classe	[C I]
I_{18}	1,3	6,1	1	[1,1]	I_{15}	5,7	7,2	2	[2,0]
I_8	1,6	3,1	1	[2,0]	I_{20}	5,4	7,4	2	[0,0]
I_3	1,8	3,9	1	[1,0]	I_5	5,5	7,2	2	[0,0]
I_7	1,5	3,5	1	[2,0]	I_{13}	1,3	3,3	1	[1,0]
I_1	1,1	3,2	1	[1,0]	I_{17}	1,3	3,1	1	[0,0]
I_{11}	5,3	7,5	2	[3,0]	I_4	5,5	7,8	2	[0,0]
I_{12}	5,7	7,8	2	[2,0]	I_{10}	5,8	7,1	2	[0,0]
I_{16}	5,2	7,3	2	[1,0]	I_{19}	6,7	7,9	2	[0,0]
I_2	5,3	7,1	2	[2,0]	I_9	5,1	5,9	2	[0,0]
I_6	2,0	4,0	1	[0,0]	I_{14}	1,3	3,7	1	[0,0]

2.5.4 Os Demais Algoritmos da Família IBL: IB2, IB3, IB4 e IB5

Os algoritmos IB1, IB2 e IB3 foram apresentados e discutidos em uma mesma publicação *i.e.,* [Aha *et al.* 1991]. O algoritmo IB2 foi proposto como uma variante do IB1 que tem por objetivo reduzir o volume de dados armazenados. Devido ao foco do algoritmo ser redução do volume de dados armazenados, que é o objetivo desta pesquisa, tal algoritmo é considerado em detalhes no Capítulo 4.

O algoritmo IB3 foi proposto para reduzir a sensitividade a dados com ruído, exibida pelo IB2. Faz isso por meio da inclusão de um teste que determina se uma dada instância será relevante futuramente ou se representa apenas ruído. Um registro de desempenho associado a cada instância armazenada é mantido. Apesar do IB3 ser uma variação do IB2 e, portanto, ter foco em redução de armazenamento, tem também foco na redução da sensitividade a dados com ruído. A redução de armazenamento implementada pelo IB3 é a mesma do IB2 e, portanto, esse algoritmo não será abordado neste trabalho.

O algoritmo IB4 foi proposto em [Aha 1992] e pode ser considerado uma extensão do IB3 que busca contornar a sensibilidade a atributos irrelevantes do IB3, 'aprendendo' a relevância dos atributos independentemente para cada conceito e usando essa informação como peso na função de similaridade que usa [Santos & Nicoletti 1997].

O IB5 foi uma proposta para o estudo da introdução de novos atributos, em uma situação de aprendizado baseada em instâncias que, apesar de importante, não é o foco deste trabalho de pesquisa. Ambos, IB4 e IB5, não são abordados posteriormente neste trabalho por terem outros focos, que não a redução em armazenamento. E ambos, de certa forma, podem ser considerados extensões do IB2, que é abordado em detalhes no Capítulo 4.

Capítulo 3
O Processo de Redução do Volume de Instâncias em ABI

3.1 Considerações Iniciais

No que segue são apresentados e discutidos alguns aspectos relevantes relacionados àqueles algoritmos ABI que, também, têm como objetivo a redução do volume de instâncias armazenadas. Tais aspectos são discutidos em inúmeras referências bibliográficas, particularmente [Aha *et al.* 1991], [Aha 1992], [Wilson & Martinez 2000] e [Brighton & Mellish 2002].

3.2 Aspectos Envolvidos no Processo de Redução do Volume de Instâncias

Quando do projeto de um algoritmo que implementa redução do volume de instâncias, uma importante escolha deve ser feita entre modificar as instâncias, usando uma nova representação ou, então, reter um subconjunto do conjunto original de instâncias.

Alguns algoritmos, tal como o EACH (*Exemplar-Aided Constructor of Hyperrectangles*) [Salzberg 1991], que implementa a teoria NGE (Nested Generalized Exemplar Learning) [Salzberg 1991] [Wettschereck 1994], [Wettschereck & Dietterich 1995], modificam e generalizam as instâncias de treinamento representando-as como hiperretângulos com faces paralelas aos eixos de coordenadas; hyperretângulos podem também ser facilmente representados como regras [Domingos 1995], [Domingos 1996].

Com relação ao aspecto *representação do conceito*, instâncias referenciadas como protótipos (ou representantes) podem representar um grupo de instâncias, mesmo que a instância protótipo tenha sido artificialmente criada (*i.e.*, não existe uma instância sua correspondente, no conjunto original de instâncias).

Algoritmos ABI buscam resolver o problema da redução do volume de instâncias por meio da identificação de um subconjunto apropriado do conjunto original de instâncias. Um problema associado à essa solução é a possibilidade de inexistência de instâncias precisamente localizadas, de maneira a refletirem uma descrição precisa e

concisa do conceito representado pelas instâncias. Por outro lado, tal precisão e concisão podem ser obtidas por meio do uso de protótipos, uma vez que podem ser criados artificialmente.

Dado um conjunto original de instâncias TR (conjunto de treinamento), a busca por um subconjunto $S_{TR} \subseteq TR$ que possa substituir TR pode acontecer usando esquema *incremental*, *decremental* ou *batch.*

Um *esquema incremental* é iniciado com $S_{TR} = \varnothing$ e, então, prossegue adicionando, uma por vez, cada uma das instâncias de TR ao conjunto S_{TR}, desde que a instância em questão satisfaça um determinado critério. Obviamente a ordem em que as instâncias de TR são consideradas tem um grande impacto no resultado final; as primeiras instâncias, por exemplo, podem ter uma probabilidade de serem incluídas em S_{TR} bem diferente que teriam, se tivessem sido consideradas depois. Via de regra em um esquema incremental as instâncias de TR são escolhidas randomicamente para serem submetidas ao teste de satisfação do critério; isso se deve ao fato de que um algoritmo incremental deve ser capaz de tratar novas instâncias à medida que elas se tornam disponíveis.

Como apontado em [Wilson & Martinez 2000] algumas vantagens do esquema incremental são: (a) possibilitar que, mesmo depois que o treinamento tenha terminado, novas instâncias possam ainda ser consideradas para participarem de $S_{TR,}$ usando o mesmo critério usado anteriormente; (b) algoritmos incrementais podem ser mais rápidos e usarem menos armazenamento durante o aprendizado do que aqueles não incrementais, uma vez que podem ignorar algumas das instâncias descartadas quando adicionando outras. Assim, durante o treinamento, ao invés do tempo envolvido e o armazenamento serem da ordem de $O(n^2)$ e $O(n)$, respectivamente, eles podem ser de $O(ns)$ e $O(s)$, respectivamente, em que n é o número de instâncias de treinamento e s é o número de instâncias em S_{TR}. Por outro lado a principal desvantagem desse esquema se deve à sua sensibilidade à ordem em que as instâncias são consideradas; as decisões iniciais são baseadas em pouca informação e são passíveis de erros até que mais informação esteja disponível.

Um *esquema decremental* é iniciado com $S_{TR} = TR$ e, então, parte para a remoção de instâncias de $S_{TR.}$ Novamente, a ordem em que as instâncias são consideradas no processo é importante mas, diferentemente da busca incremental, todas as instâncias de treinamento estão sempre disponíveis para inspeção, de maneira que uma busca pode ser feita para determinar qual a melhor instância a ser removida, a cada passo do

processo. Uma desvantagem do esquema decremental é que, frequentemente, tem um custo computacional mais elevado do que aquele do esquema incremental. Para determinar a vizinha mais próxima de uma instância em T, n cálculos de distância precisam ser feitos, enquanto que no esquema incremental existe um número bem menor (que n) de instâncias em S_{TR}, o que torna o cálculo de uma vizinha mais próxima, em S_{TR}, computacionalmente menos custoso.

No *esquema batch* o processo associado verifica se cada instância satisfaz o critério para a sua remoção, antes da remoção de qualquer uma delas. As que satisfizerem o critério, são removidas de uma vez. Isso pode aliviar o algoritmo de ter que, constantemente, atualizar as listas de instâncias mais próximas e outras informações, quando da remoção individual de instâncias. Por outro lado, entretanto, o esquema *batch* pode ter efeitos colaterais indesejáveis. Por exemplo, em uma situação em que o critério para remoção de instâncias seja: "*remover uma instância se ela pertence à mesma classe que k de suas vizinhas mais próximas*", pode acontecer a remoção de um grupo inteiro de instâncias de uma mesma classe, caso não existam instâncias de outra classe por perto. De maneira similar ao esquema decremental, o esquema *batch* também sofre com um aumento de complexidade em tempo, quando comparado com o esquema incremental.

Um outro aspecto relevante relacionado às técnicas que promovem a redução do volume de instâncias diz respeito à *região do espaço conceitual que a técnica em questão privilegia*, quando da retenção de instâncias: algumas podem buscar a retenção de instâncias da fronteira, outras as instâncias centrais ou, então, outros subconjuntos de instâncias. A intuição que promove a retenção de instâncias da fronteira é a que instâncias internas não afetam as fronteiras de decisão tanto quanto as instâncias da fronteira e, assim, podem ser removidas com pouco impacto na classificação.

Outros algoritmos, entretanto, removem instâncias da fronteira que têm ruídos ou que não têm a mesma classe que as instâncias suas vizinhas. Tais algoritmos não removem instâncias internas que não necessariamente contribuem com a fronteira de decisão (i.e., a fronteira que separa classes distintas). Como muitas vezes é necessário um número razoável de instâncias para definir completamente a fronteira, alguns algoritmos retém instâncias centrais, com o objetivo de utilizar as instâncias que são mais típicas de uma determinada classe para classificar instâncias que estão perto delas. Isso pode ter um grande impacto na fronteira de decisão uma vez que as fronteiras de

decisão depende não apenas onde as instâncias de uma classe estão posicionadas, mas também, onde as instâncias de outras classes estão posicionadas.

Um outro aspecto a ser considerado é o que envolve a *função de distância*, utilizada para identificar quais as instâncias estão mais perto de uma determinada instância; a escolha de tal função tem um grande impacto no resultado de um algoritmo ABI. Via de regra a distância euclidiana é empregada que, por essa razão, será a função utilizada neste trabalho. Existem, entretanto, inúmeros estudos que têm por foco a investigação detalhada dos impactos da escolha de tal função. Detalhes e resultados podem ser consultados em [Wilson & Martinez 1977a].

3.3 Estratégias de Avaliação de Algoritmos de Redução

Vários critérios podem ser usados para comparar o desempenho de algoritmos para a redução do volume de instâncias. Wilson e Martinez em [Wilson & Martinez 2000] consideram seis critérios que serão brevemente abordados nesta seção: (1) redução de armazenamento; (2) aumento de velocidade (durante classificação); (3) tolerância a ruídos nos dados; (4) precisão de generalização; (5) exigências de tempo (durante o aprendizado) e (6) incrementabilidade.

(1) Redução de armazenamento: é o principal objetivo de algoritmos de redução e, portanto, um dos critérios mais relevantes para avaliar um algoritmo de redução. Obviamente se representações alternativas forem utilizadas, um aumento no tamanho da nova representação deve ser levado em conta, juntamente com a redução do número de instâncias armazenadas.

(2) Aumento de velocidade (durante classificação): o aumento de velocidade durante o processo de classificação é consequência direta da redução do número de instâncias armazenadas, uma vez que implica um número menor de possíveis comparações da instância a ser classificada com as instâncias armazenadas. É importante lembrar que quando do uso de representações mais elaboradas, hiperretângulos por exemplo, apesar do número de comparações tender a ser bem menor, é fato que tais comparações podem requerer um tempo computacional maior, devido ao fato da representação (hiperretângulo, no caso), ser mais elaborada do que a representação de uma instância pontual.

(3) Tolerância a ruídos: algoritmos de redução podem diferir entre si com relação à maneira como lidam com ruídos nos dados. Como discutido em [Wilson & Martinez 2000], com relação à ruídos na informação que representa as classes de instâncias, dois problemas podem ocorrer: (a) poucas instâncias serão removidas do conjunto de treinamento uma vez que muitas instâncias são necessárias para manter as fronteiras (com ruído) de decisão; (b) a precisão na generalização pode sofrer impacto, especialmente se instâncias com ruídos são retidas, enquanto instâncias representativas são removidas. Em situações como essa, quando da classificação de novas instâncias, o conjunto de treinamento reduzido pode ter uma acurácia muito menor do que aquela obtida com o conjunto de treinamento completo.

(4) Precisão de generalização: um algoritmo é considerado bem-sucedido se for capaz de reduzir significativamente o volume do conjunto de instâncias armazenadas, sem provocar uma redução significativa na acurácia do processo de classificação. Particularmente em situações em que instâncias com ruídos são removidas e, também, quando as fronteiras de decisão são suavizadas, de maneira a se aproximarem mais da função que define a correspondência entre instâncias e classes, do que da distribuição das instâncias do conjunto de instâncias considerado.

(5) Exigências de tempo (durante o aprendizado): o processo de aprendizado (i.e., a fase de treinamento) é realizada uma única vez considerando o conjunto de treinamento e, portanto, o tempo para sua finalização não necessariamente precisa ser extremamente curto. Entretanto, se essa fase demorar muito, o seu uso fica impraticável em aplicações em tempo real. Como algoritmos de redução de volume de instâncias são particularmente importantes em situações em que o conjunto de treinamento é bem volumoso, um limite de tempo da ordem de $O(n^2)$ é desejável (n é o número de instâncias inicial).

(6) *Incrementabilidade:* às vezes é conveniente ter um algoritmo incremental de maneira que instâncias adicionais podem ser adicionadas com o passar do tempo, assim que se tornarem disponíveis. É também possível usar um algoritmo não incremental em uma base de dados inicial e, então, seu resultado ser usado como ponto de partida para um algoritmo incremental.

Capítulo 4
O Algoritmo IB2 da Família IBL

4.1 Considerações Iniciais

Como adiantado na Seção 2.5.4, o algoritmo IB2, da família IBL, investigado neste capítulo, foi proposto com o intuito de minimizar o volume de dados de um algoritmo IBL (no caso, o IB1) [Aha *et al.* 1991]. O IB2 pode ser abordado como uma variante do IB1, que busca minimizar o volume de instâncias armazenadas pelo algoritmo original.

4.2 O Algoritmo IB2

O IB2 [Aha *et al.* 1991] usa as mesmas funções de similaridade e de classificação usadas pelo IB1, como definidas na Seção 2.5. A diferença entre o IB1 e o IB2 está no atualizador da descrição do conceito que, no IB2, durante o processo de armazenamento do conjunto de treinamento, armazena apenas as instâncias classificadas incorretamente (por aquelas instâncias que já foram armazenadas). O pseudocódigo do algoritmo está descrito em Algoritmo 4.1.

Como apontado em [Santos & Nicoletti 1997] a estratégia de armazenamento adotada pelo IB2 foi motivada pela intuição de que a maioria das instâncias classificadas incorretamente encontra-se na "fronteira" do conceito e, de certa forma, o delimitam. Ambos os algoritmos, IB1 e IB2, não removem qualquer instância da descrição do conceito, depois que tal descrição da instância tenha sido armazenada.

Como discutido em [Aha *et al.* 1991], o IB2 diminui consideravelmente a necessidade de armazenamento o que, entretanto, o torna mais sensível à presença de ruído nos dados de treinamento, como consequência do armazenamento de instâncias classificadas incorretamente.

Como instâncias com ruídos são, quase sempre, classificadas incorretamente, tais instâncias acabam por participar da descrição do conceito. Em domínios com alta incidência de dados com ruídos, o volume de tais dados na expressão final do conceito acaba sendo expressivo, o que afeta de forma negativa o desempenho do sistema na fase de classificação. Em situações de aprendizado automático em que dados com ruídos não comparecem no conjunto de treinamento, o IB2 reduz significativamente a necessidade

de armazenamento requerida pelo IB1 e os dois algoritmos têm precisão de classificação similar.

Os autores ainda apontam que, embora o algoritmo IB2 possa reduzir significantemente a quantidade de instâncias armazenadas, ele sacrifica a precisão de classificação, em situações em que o conjunto de treinamento possui instâncias com ruídos. O pseudocódigo do algoritmo IB2 está descrito no Algoritmo 4.1 exatamente como apresentado em [Aha *et al.* 1991]. Com o intuito de refinar o Algoritmo 4.1, considerando que nele a descrição do conceito não é inicializada, ele foi ligeiramente alterado, com vistas a uma melhor compreensão de seu funcionamento, com a inclusão das linhas 4–6. As linhas 12 e 14 sofreram alterações com a substituição da variável classificação pelas variáveis classificação_correta e classificação_incorreta, para deixar claro a atualização dos placares correto e incorreto, na versão apresentada em Algoritmo 4.2. Também, como a função empregada nas implementações foi a distância euclidiana (comumente usada por ABI), a função implementada por tal distância é a dissimilaridade, no sentido que quanto maior a distância euclidiana entre duas instâncias, maior é a dissimilaridade entre elas. Quando do emprego de dissimilaridade, a busca é pela instância que apresenta a menor dissimilaridade com relação à instância sendo considerada para participar do conceito.

```
procedure ib2
input: TR {conjunto de treinamento}
begin
  DC ← ∅
  for each x ∈ TR do
   begin
      for each y ∈ DC do sim[y] ← similaridade(x,y)
      y_max ← algum y ∈ DC com sim[y] maximal
      if classe(x) = classe(y_max)
       then classificação ← correta
       else begin
              classificação ← incorreta
              DC ← DC ∪ {x}
            end
   end for
return DC
end procedure
```

Algoritmo 4.1 Pseudocódigo do IB2, como apresentado em [Aha *et al.*1991], em que TR: conjunto de treinamento e DC: descrição do conceito.

```
1. procedure ib2
2. input: TR {conjunto de treinamento}
3. begin
4.   z ← random_selection(TR)
5.   TR ← TR −{z}
6.   DC ← {z}
7.   for each x ∈ TR do
8.    begin
9.         for each y ∈ DC do sim[y] ← dissimilaridade(x,y)
10.         y_min ← algum y ∈ DC com dissim[y] minimal
11.         if classe(x) = classe(y_min)
12.           then classificação_correta ← classificação _correta +1
13.          else begin
14.                 classificação_incorreta ← incorreta_incorreta + 1
15.                  DC ← DC ∪ {x}
16.                end
17.   end for
18. return DC
19. end procedure
```

Algoritmo 4.2 Variante do pseudocódigo do IB2, proposta neste trabalho, baseada no pseudocódigo apresentado em [Aha *et al.*1991] (Figura 4.1), com inicialização da DC (descrição do conceito).

4.3 Primeiro Exemplo de Uso do IB2

Nesta seção será apresentado um exemplo do uso do IB2 (Algoritmo 4.2), que recebe como conjunto de treinamento o conjunto de instâncias da Tabela 2.1, da Seção 2.3.2; para facilitar a leitura, aquela tabela está reescrita como Tabela 4.1, na próxima página.

Considere que as instâncias de dados são fornecidas ao IB2 de maneira sequencial, como aparecem no conjunto de treinamento. Suponha que a função *random_selection*() do Algoritmo 4.2 retorna a primeira instância do conjunto de treinamento, que é a instância I_1 = (1,1 3,2 1) *i.e.*, o ponto de coordenadas (1,1 3,2) que pertence à classe 1. Essa instância passa a fazer parte da descrição do conceito (*i.e.*, é armazenada). A partir desse ponto, em uma abordagem sequencial, a próxima instância é a I_2 = (5,3 7,1 1) *i.e.*, o ponto de coordenadas (5,3 7,1), que pertence à classe 2.

Tabela 4.1 Conjunto de treinamento contendo 20 instâncias de dados, sendo 9 de classe 1 e 11 de classe 2.

#ID	X	Y	Classe	#ID	X	Y	Classe
I_1	1,1	3,2	1	I_{11}	5,3	7,5	2
I_2	5,3	7,1	2	I_{12}	5,7	7,8	2
I_3	1,8	3,9	1	I_{13}	1,3	3,3	1
I_4	5,5	7,8	2	I_{14}	1,3	3,7	1
I_5	5,5	7,2	2	I_{15}	5,7	7,2	2
I_6	2,0	4,0	1	I_{16}	5,2	7,3	2
I_7	1,5	3,5	1	I_{17}	1,3	3,1	1
I_8	1,6	3,1	1	I_{18}	1,3	6,1	1
I_9	5,1	5,9	2	I_{19}	6,7	7,9	2
I_{10}	5,8	7,1	2	I_{20}	5,4	7,4	2

O algoritmo então calcula a similaridade entre $I_1 \in DC$ e I_2 (que será a única similaridade passível de ser calculada, considerando que existe apenas um elemento em DC) e, portanto, $y_{min} = I_1$. Como as classes de I_1 e I_2 são diferentes, a classificação, feita por I_1, da instância I_2, foi incorreta e, portanto, a instância I_2 passa a fazer parte de DC e a pontuação da instância I_1 passa a acusar uma classificação incorreta e zero correta i.e., [C:0 I:1].

A próxima instância que é entrada para o IB2 é a instância I_3 = (1,8 3,9 1) ou seja, o ponto (1,8 3,9), de classe 1. Como DC = {I_1, I_2}, a dissimilaridade da nova instância (I_3) deve ser calculada com relação a ambas as instâncias, I_1 e I_2. A Tabela 4.2 mostra os cálculos.

$$\text{dissimilaridade}(x_i,x_j) = \text{distância}(x_i,x_j) \qquad (5.1)$$

Tabela 4.2 Dissimilaridade entre a instância I_3 e as instâncias I_1 e I_2, em que d = distância.

$d(I_1,I_3) = d((1,1\ \ 3,2), (1,8\ \ 3,9)) = SQRT((1,1 - 1,8)^2 + (3,2 - 3,9)^2 = SQRT(0,98) = 0,9899$
$d(I_2,I_3) = d((5,3\ \ 7,1), (1,8\ \ 3,9)) = SQRT((5,3 - 1,8)^2 + (7,1 - 3,9)^2 = SQRT(22.49) = 4,7423$

Como a dissimilaridade entre I_3 e I_1 é a menor, I_3 é classificada com a classe de I_1 ou seja, com classe 1. E a contabilização de classificações corretas de I_1 cresce em 1, ficando [C:1 I:1], Como I_3 foi corretamente classificada por I_1, I_3 não é incluída na descrição do conceito, DC.

Efetuando o cálculo da dissimilaridade para todo o conjunto de treinamento, apenas as instâncias I_1 e I_2 irão fazer parte de DC, como pode ser observado na Tabela 4.3. Todas as instâncias, desde I_3 até I_{20}, irão ser descartadas da DC, uma vez que o menor valor de dissimilaridade entre cada uma delas e sua mais próxima instância em DC, é sempre uma instância de mesma classe que aquela sendo classificada. Quando o algoritmo finaliza, os registros de classificação associados a cada uma das instâncias

que participa da descrição do conceito das instâncias são: I_1 = [C:8 I:1] e I2 = [C:10 I:0].

Tabela 4.3 Matriz de valores de dissimilaridade considerando todas as instâncias do conjunto de treinamento.

	$I_1(1)$	$I_2(2)$	$I_3(1)$	$I_4(2)$	$I_5(2)$	$I_6(1)$	$I_7(1)$	$I_8(1)$	$I_9(2)$	$I_{10}(2)$	$I_{11}(2)$	$I_{12}(2)$	$I_{13}(1)$	$I_{14}(1)$	$I_{15}(2)$	$I_{16}(2)$	$I_{17}(1)$	$I_{18}(1)$	$I_{19}(2)$	$I_{20}(2)$
$I_1(1)$	0,00																			
$I_2(2)$	5,73	0,00																		
$I_3(1)$	0,99	4,74	0,00																	
$I_4(2)$	6,37	0,73	5,38	0,00																
$I_5(2)$	5,95	0,22	4,96	0,60	0,00															
$I_6(1)$	1,20	4,53	0,22	5,17	4,74	0,00														
$I_7(1)$	0,50	5,23	0,50	5,87	5,45	0,71	0,00													
$I_8(1)$	0,51	5,45	0,82	6,11	5,66	0,98	0,41	0,00												
$I_9(2)$	4,83	1,22	3,86	1,94	1,36	3,64	4,33	4,88	0,00											
$I_{10}(2)$	6,11	0,50	5,12	0,76	0,32	4,90	5,61	5,80	1,39	0,00										
$I_{11}(2)$	6,01	0,40	5,02	0,36	0,36	4,81	5,52	5,75	1,61	0,64	0,00									
$I_{12}(2)$	6,51	0,81	5,52	0,20	0,63	5,30	6,01	6,24	1,99	0,71	0,50	0,00								
$I_{13}(1)$	0,22	5,52	0,78	6,16	5,73	0,99	0,28	0,36	4,60	5,89	5,80	6,29	0,00							
$I_{14}(1)$	0,54	5,25	0,54	5,87	5,47	0,76	0,28	0,67	4,39	5,64	5,52	6,01	0,40	0,00						
$I_{15}(2)$	6,10	0,41	5,11	0,63	0,20	4,89	5,60	5,80	1,43	0,14	0,50	0,60	5,88	5,62	0,00					
$I_{16}(2)$	5,80	0,22	4,81	0,58	0,32	4,60	5,30	5,53	1,40	0,63	0,22	0,71	5,59	5,31	0,51	0,00				
$I_{17}(1)$	0,22	5,66	0,94	6,30	5,87	1,14	0,45	0,30	4,72	6,02	5,95	6,44	0,20	0,60	6,01	5,73	0,00			
$I_{18}(1)$	2,91	4,12	2,26	4,53	4,34	2,21	2,61	3,01	3,81	4,61	4,24	4,72	2,80	2,40	4,54	4,08	3,00	0,00		
$I_{19}(2)$	7,31	1,61	6,33	1,20	1,39	6,11	6,81	7,00	2,56	1,20	1,46	1,00	7,09	6,84	1,22	1,62	7,22	5,69	0,00	
$I_{20}(2)$	6,01	0,32	5,02	0,41	0,22	4,81	5,52	5,74	1,53	0,50	0,14	0,50	5,80	5,52	0,36	0,22	5,94	4,30	1,39	0,00

Apenas com vistas à uma comparação com relação à expressão final do conceito induzido, a Figura 4.1 apresenta graficamente a expressão do conceito induzida pelo IB1, *i.e.*, todo o conjunto de treinamento, e a Figura 4.2, a expressão do conceito induzida pelo IB2, constituída por apenas duas instâncias, I_1 e I_2, que classificam todas as demais corretamente. A Tabela 4.4 traz os registros de desempenho associados a cada instância que participa da descrição do conceito induzido pelo IB1 e a Tabela 4.5 traz os registros de desempenho das duas instâncias que participam da descrição do conceito induzido pelo IB2.

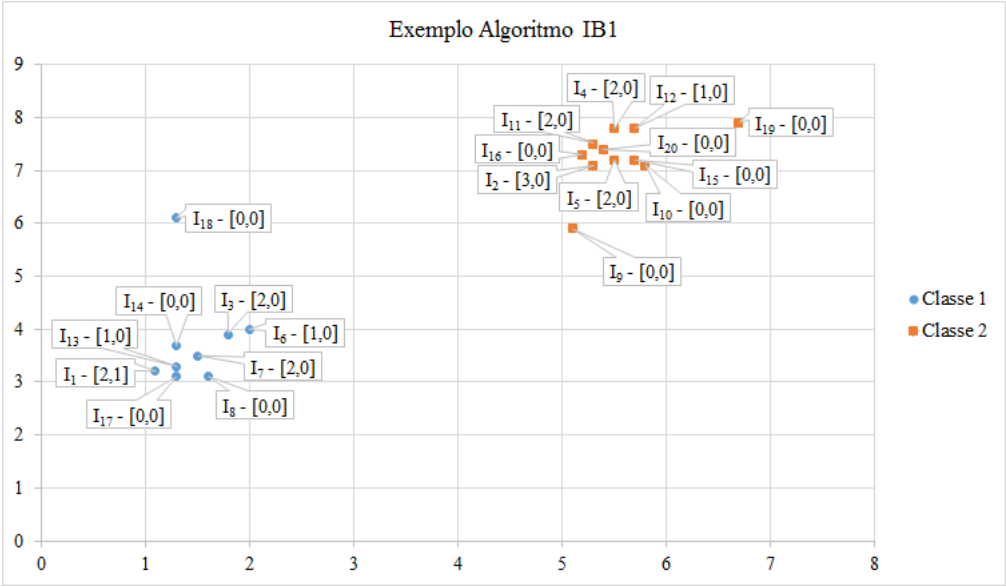


Figura 4.1 Instâncias armazenadas em DC utilizando o algoritmo IB1 e seus respectivos índices de classificação [Corretas,Incorretas].

Tabela 4.4 Conceito induzido pelo IB1, a partir do conjunto de instâncias da Tabela 4.1 com o registro de desempenho associado a cada instância.

#ID	X	Y	Classe	[C	I]	#ID	X	Y	Classe	[C	I]
I_1	1,1	3,2	1	[2	1]	I_{11}	5,3	7,5	2	[2	0]
I_2	5,3	7,1	2	[3	0]	I_{12}	5,7	7,8	2	[1	0]
I_3	1,8	3,9	1	[2	0]	I_{13}	1,3	3,3	1	[1	0]
I_4	5,5	7,8	2	[2	0]	I_{14}	1,3	3,7	1	[0	0]
I_5	5,5	7,2	2	[2	0]	I_{15}	5,7	7,2	2	[0	0]
I_6	2,0	4,0	1	[1	0]	I_{16}	5,2	7,3	2	[0	0]
I_7	1,5	3,5	1	[2	0]	I_{17}	1,3	3,1	1	[0	0]
I_8	1,6	3,1	1	[0	0]	I_{18}	1,3	6,1	1	[0	0]
I_9	5,1	5,9	2	[0	0]	I_{19}	6,7	7,9	2	[0	0]
I_{10}	5,8	7,1	2	[0	0]	I_{20}	5,4	7,4	2	[0	0]

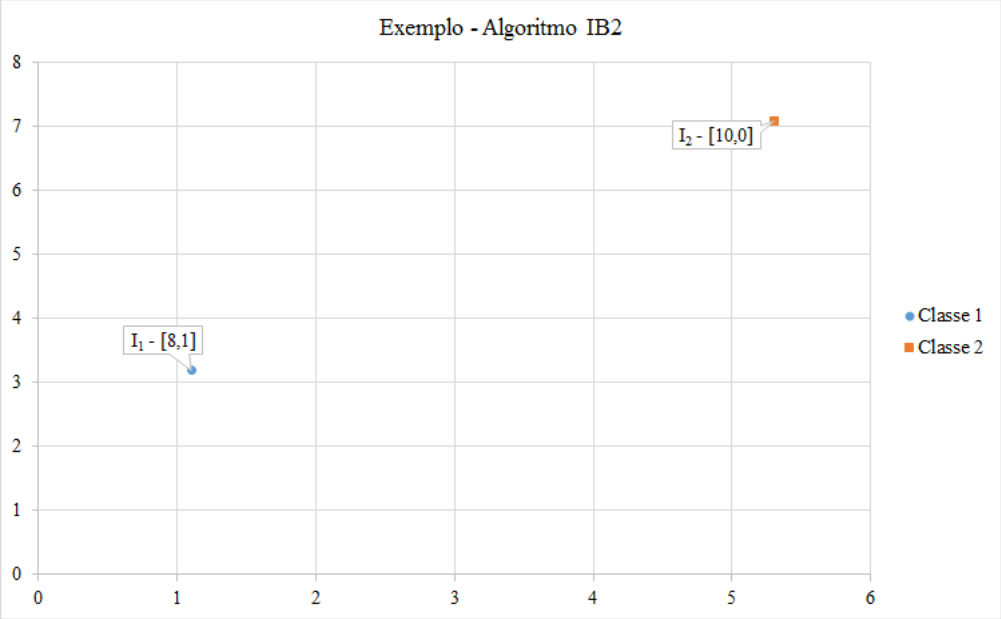


Figura 4.2 Instâncias armazenadas em DC utilizando o algoritmo IB2 e seus respectivos índices de classificação [Corretas,Incorretas].

Tabela 4.5 Conceito induzido pelo IB2, a partir do conjunto de instâncias da Tabela 4.1, com o registro de desempenho associado a cada instância.

#ID	X	Y	Classe	[C I]
I_1	1,1	3,2	1	[8 1]
I_2	5,3	7,1	2	[10 0]

4.4 Segundo Exemplo de Uso do IB2

O exemplo que segue mostra uma segunda situação de uso do IB2, em que o conjunto de treinamento tem também 20 instâncias de dados bidimensionais, divididas em duas classes, sendo 9 de classe 1 e 11 de classe 2, como pode ser evidenciado na Tabela 4.6.

Tabela 4.6 Conjunto de treinamento contendo 20 instâncias de dados, sendo 9 de classe 1 e 11 de classe 2.

#ID	X	Y	Classe	#ID	X	Y	Classe
I_1	2,0	2,0	1	I_{11}	2,0	10,0	1
I_2	4,0	3,0	2	I_{12}	4,0	11,0	2
I_3	3,0	4,0	1	I_{13}	5,0	5,0	2
I_4	3,0	6,0	2	I_{14}	1,0	12,0	1
I_5	1,0	4,0	1	I_{15}	3,0	12,0	2
I_6	1,0	7,0	1	I_{16}	1,0	3,0	1
I_7	2,0	8,0	2	I_{17}	6,0	4,0	2
I_8	1,0	9,0	1	I_{18}	4,0	7,0	2
I_9	3,0	9,0	2	I_{19}	1,0	1,0	1
I_{10}	5,0	8,0	2	I_{20}	5,0	2,0	2

Seguindo o exemplo anterior mostrado na Seção 4.3 e utilizando a distância euclidiana como função de dissimilaridade, o algoritmo IB2 alimentado com os dados da Tabela 4.6 armazenou na Descrição do Conceito 9 das 20 instâncias do conjunto, o que representa uma redução de 55% do espaço de armazenamento, quando comparado ao armazenamento de todas as instâncias, feito pelo IB1. A Tabela 4.7 apresenta os valores de distância entre pares de instâncias, para todas as instâncias do conjunto mostrado na Tabela 4.6. Note que as instâncias I_5, I_9–I_{11} e I_{14}–I_{20} não foram armazenadas em DC pois foram classificadas corretamente pelas instâncias da DC que lhes foram mais próximas.

Tabela 4.7 Cálculo da distância entre as instâncias do conjunto de instâncias da Tabela 4.6, utilizando o algoritmo IB2.

	$I_1(1)$	$I_2(2)$	$I_3(1)$	$I_4(2)$	$I_5(1)$	$I_6(1)$	$I_7(2)$	$I_8(1)$	$I_9(2)$	$I_{10}(2)$	$I_{11}(1)$	$I_{12}(2)$	$I_{13}(2)$	$I_{14}(1)$	$I_{15}(2)$	$I_{16}(1)$	$I_{17}(2)$	$I_{18}(2)$	$I_{19}(1)$	$I_{20}(2)$
$I_1(1)$	0,00																			
$I_2(2)$	2,24	0,00																		
$I_3(1)$	2,24	1,41	0,00																	
$I_4(2)$	4,12	3,16	2,00	0,00																
$I_5(1)$	2,24	3,16	2,00	2,83	0,00															
$I_6(1)$	5,10	5,00	3,61	2,24	3,00	0,00														
$I_7(2)$	6,00	5,39	4,12	2,24	4,12	1,41	0,00													
$I_8(1)$	7,07	6,71	5,39	3,61	5,00	2,00	1,41	0,00												
$I_9(2)$	7,07	6,08	5,00	3,00	5,39	2,83	1,41	2,00	0,00											
$I_{10}(2)$	6,71	5,10	4,47	2,83	5,66	4,12	3,00	4,12	2,24	0,00										
$I_{11}(1)$	8,00	7,28	6,08	4,12	6,08	3,16	2,00	1,41	1,41	3,61	0,00									
$I_{12}(2)$	9,22	8,00	7,07	5,10	7,62	5,00	3,61	3,61	2,24	3,16	2,24	0,00								
$I_{13}(2)$	4,24	2,24	2,24	2,24	4,12	4,47	4,24	5,66	4,47	3,00	5,83	6,08	0,00							
$I_{14}(1)$	10,05	9,49	8,25	6,32	8,00	5,00	4,12	3,00	3,61	5,66	2,24	3,16	8,06	0,00						
$I_{15}(2)$	10,05	9,06	8,00	6,00	8,25	5,39	4,12	3,61	3,00	4,47	2,24	1,41	7,28	2,00	0,00					
$I_{16}(1)$	1,41	3,00	2,24	3,61	1,00	4,00	5,10	6,00	6,32	6,40	7,07	8,54	4,47	9,00	9,22	0,00				
$I_{17}(2)$	4,47	2,24	3,00	3,61	5,00	5,83	5,66	7,07	5,83	4,12	7,21	7,28	1,41	9,43	8,54	5,10	0,00			
$I_{18}(2)$	5,39	4,00	3,16	1,41	4,24	3,00	2,24	3,61	2,24	1,41	3,61	4,00	2,24	5,83	5,10	5,00	3,61	0,00		
$I_{19}(1)$	1,41	3,61	3,61	5,39	3,00	6,00	7,07	8,00	8,25	8,06	9,06	10,44	5,66	11,00	11,18	2,00	5,83	6,71	0,00	
$I_{20}(2)$	3,00	1,41	2,83	4,47	4,47	6,40	6,71	8,06	7,28	6,00	8,54	9,06	3,00	10,77	10,20	4,12	2,24	5,10	4,12	0,00

Novamente, com vistas à uma visualização gráfica do volume de instâncias armazenado pelo IB2, é mostrado na Figura 4.3 a representação gráfica da DC obtida utilizando o algoritmo IB1 (*i.e.*, todo o conjunto de treinamento), em que cada instância é acompanhada de seus índices de classificação. A Figura 4.4 apresenta a representação gráfica da DC obtida utilizando o algoritmo IB2, em que cada instância é acompanhada de seus respectivos índices de classificação.

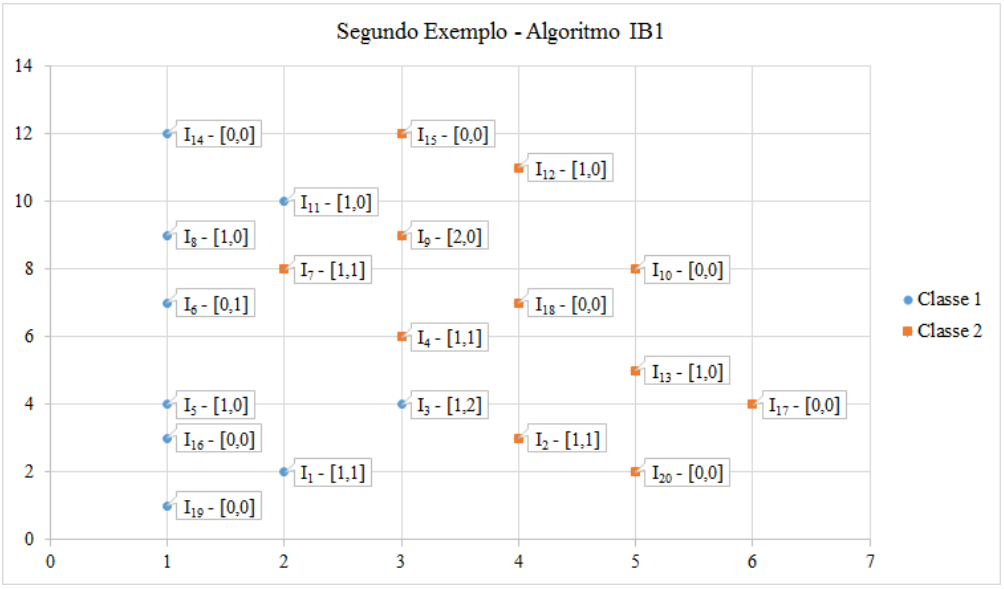


Figura 4.3 Representação gráfica das instâncias que participam da descrição do conceito induzida pelo IB1 (i.e., todo o conjunto de treinamento), nos dados da Tabela 4.6. Cada uma das instâncias participantes está acompanhada de seu registro de desempenho.

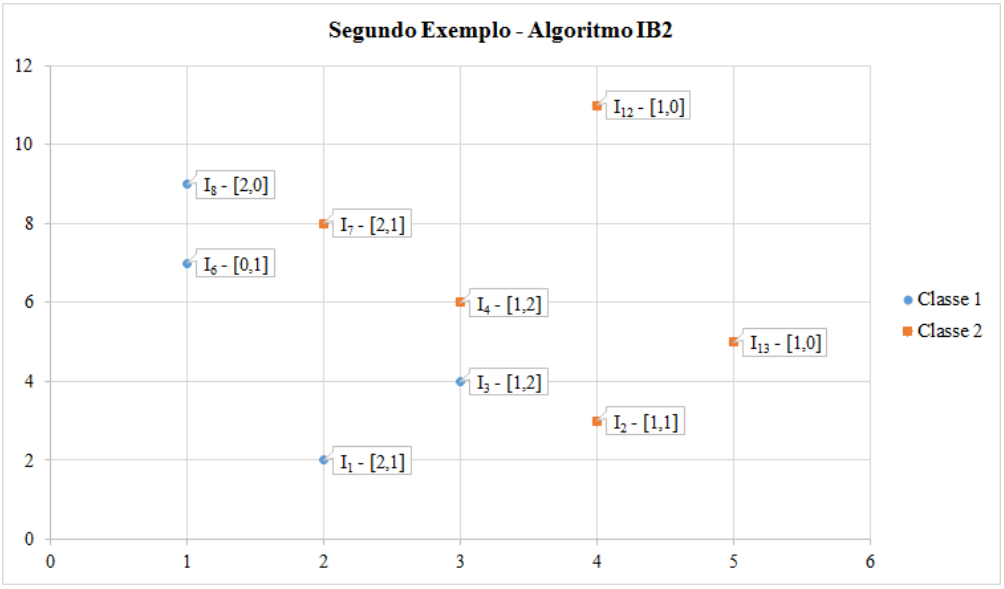


Figura 4.4 Representação gráfica das instâncias que participam da descrição do conceito induzida pelo IB2, nos dados da Tabela 4.6. Cada uma das instâncias participantes está acompanhada de seu registro de desempenho.

Ainda como continuação desse exemplo, é mostrado a seguir o resultado do uso do IB2 considerando o mesmo conjunto de dados da Tabela 4.6 mas 'embaralhados', de maneira a apresentá-los de uma maneira diferente daquela da Tabela 4.6. A Tabela 4.8 mostra o conjunto 'embaralhado' e a Figura 4.5 mostra o resultado do uso do IB2 nas instâncias da Tabela 4.8, fornecidas ao algoritmo na ordem em que comparecem na tabela *i.e.*, I_{20}, I_{19}, I_3, ..., I_{13}.

Tabela 4.8 Conjunto de treinamento da Tabela 4.6, 'embaralhado'.

#ID	X	Y	Classe	#ID	X	Y	Classe
I_{20}	5,0	2,0	2	I_7	2,0	8,0	2
I_{19}	1,0	1,0	1	I_1	2,0	2,0	1
I_3	3,0	4,0	1	I_{11}	2,0	10,0	1
I_{17}	6,0	4,0	2	I_{10}	5,0	8,0	2
I_2	4,0	3,0	2	I_{18}	4,0	7,0	2
I_5	1,0	4,0	1	I_9	3,0	9,0	2
I_{15}	3,0	12,0	2	I_6	1,0	7,0	1
I_4	3,0	6,0	2	I_{16}	1,0	3,0	1
I_8	1,0	9,0	1	I_{12}	4,0	11,0	2
I_{14}	1,0	12,0	1	I_{13}	5,0	5,0	2

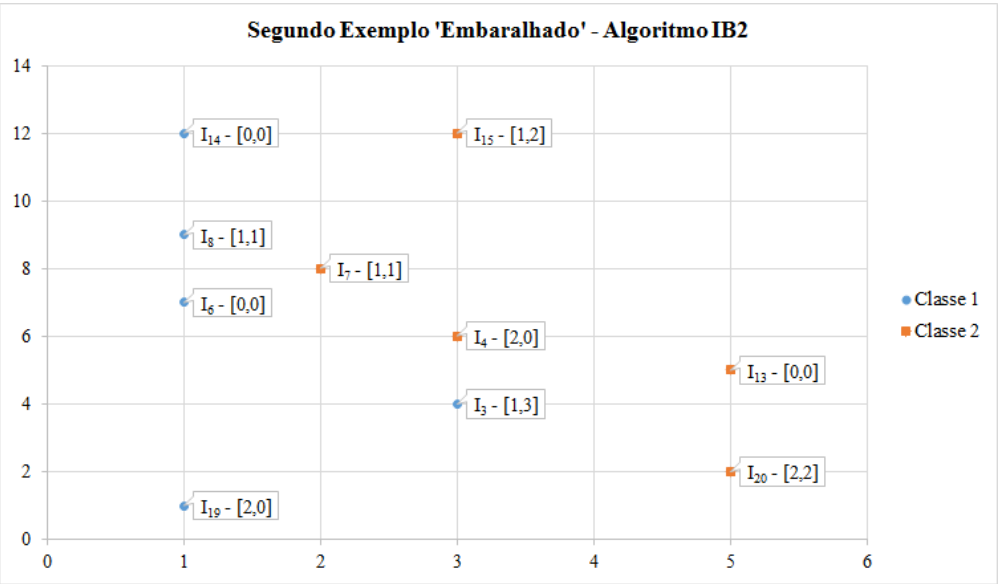


Figura 4.5 Representação gráfica das instâncias que participam da descrição do conceito induzida pelo IB2, nos dados da Tabela 4.7, que é uma reordenação dos dados da Tabela 4.6. Cada uma das instâncias participantes está acompanhada de seu registro de desempenho.

4.5 Terceiro Exemplo de Uso do IB2

O terceiro exemplo de uso do IB2 considera um conjunto de 20 instâncias de dados bidimensionais, pertencentes a três classes distintas, 1, 2 e 3, como mostra a Tabela 4.9.

Tabela 4.9 Conjunto de 20 instâncias de dados pertencentes a três classes distintas, com as seguintes distribuições: 1(7), 2(8) e 3(5).

#ID	X	Y	Classe	#ID	X	Y	Classe
I_1	4,0	2,9	2	I_{11}	1,2	1,4	3
I_2	6,2	5,0	1	I_{12}	9,0	9,0	2
I_3	9,3	11,0	3	I_{13}	7,8	5,0	1
I_4	8,3	7,2	1	I_{14}	12,8	8,0	3
I_5	6,8	9,0	2	I_{15}	9,6	6,0	1
I_6	4,2	11,0	1	I_{16}	6,7	0,7	3
I_7	10,5	2,8	2	I_{17}	7,8	3,6	2
I_8	11,7	6,0	2	I_{18}	4,7	5,5	1
I_9	11,0	1,0	3	I_{19}	2,8	5,4	2
I_{10}	3,0	8,3	2	I_{20}	5,7	7,2	1

Para o uso do IB2 no conjunto da Tabela 4.9, os cálculos das distâncias entre pares de instâncias estão mostrados na Tabela 4.10. A descrição do conceito induzida pelo IB2 com relação a esse particular conjunto de instâncias contém 15 instâncias, o que reduziu em 25% o volume de armazenamento. As instâncias I_4, I_8, I_{13}, I_{18} e I_{19}, não fazem parte da Descrição do Conceito pois durante o aprendizado pelo IB2, foram classificadas corretamente pelas instâncias já participantes de DC.

Tabela 4.10 Cálculo da distância entre as instâncias de dados (Tabela 4.8) fornecidas ao IB2.

	I_1(2)	**I_2(1)**	**I_3(3)**	**I_4(1)**	**I_5(2)**	**I_6(1)**	**I_7(2)**	**I_8(2)**	**I_9(3)**	**I_{10}(2)**	**I_{11}(3)**	**I_{12}(2)**	**I_{13}(1)**	**I_{14}(3)**	**I_{15}(1)**	**I_{16}(3)**	**I_{17}(2)**	**I_{18}(1)**	**I_{19}(2)**	**I_{20}(1)**
I_1(2)	0,00																			
I_2(1)	3,04	0,00																		
I_3(3)	9,68	6,75	0,00																	
I_4(1)	6,08	3,04	3,93	0,00																
I_5(2)	6,71	4,04	3,20	2,34	0,00															
I_6(1)	8,10	6,32	5,10	5,59	3,28	0,00														
I_7(2)	6,50	4,83	8,29	4,92	7,22	10,34	0,00													
I_8(2)	8,30	5,59	5,55	3,61	5,75	9,01	3,42	0,00												
I_9(3)	7,25	6,25	10,14	6,76	9,04	12,09	1,87	5,05	0,00											
I_{10}(2)	5,49	4,60	6,85	5,41	3,86	2,95	9,30	9,00	10,83	0,00										
I_{11}(3)	3,18	6,16	12,56	9,17	9,44	10,06	9,40	11,46	9,81	7,13	0,00									
I_{12}(2)	7,89	4,88	2,02	1,93	2,20	5,20	6,38	4,04	8,25	6,04	10,89	0,00								
I_{13}(1)	4,34	1,60	6,18	2,26	4,12	7,00	3,48	4,03	5,12	5,82	7,52	4,18	0,00							
I_{14}(3)	9,74	6,80	4,24	4,08	5,59	8,64	5,50	2,09	7,12	9,30	12,91	3,45	5,41	0,00						
I_{15}(1)	6,40	3,54	5,01	1,77	4,10	7,36	3,32	2,10	5,19	6,99	9,58	3,06	2,06	3,36	0,00					
I_{16}(3)	3,48	4,33	10,62	6,69	8,30	10,60	4,34	7,29	4,31	8,45	5,54	8,61	4,44	9,20	6,04	0,00				
I_{17}(2)	3,86	2,13	7,55	3,63	5,49	8,23	2,82	4,58	4,12	6,72	6,96	5,53	1,40	6,29	3,00	3,10	0,00			
I_{18}(1)	2,69	1,58	7,17	3,98	4,08	5,52	6,40	7,02	7,74	3,28	5,39	5,54	3,14	8,00	4,93	5,20	3,64	0,00		
I_{19}(2)	2,77	3,42	8,58	5,79	5,38	5,77	8,13	8,92	9,31	2,91	4,31	7,17	5,02	9,85	6,83	6,11	5,31	2,42	0,00	
I_{20}(1)	4,62	2,26	5,23	2,60	2,11	4,09	6,51	6,12	8,16	2,92	7,34	3,76	3,04	6,65	4,08	6,58	4,17	1,89	3,41	0,00

Para facilitar uma visualização comparativa com o resultado obtido pelo IB1, a

Figura 4.6 apresenta as instâncias participantes da DC induzida pelo IB1 e a Figura 4.7 apresenta as instâncias participantes da DC induzida pelo IB2.

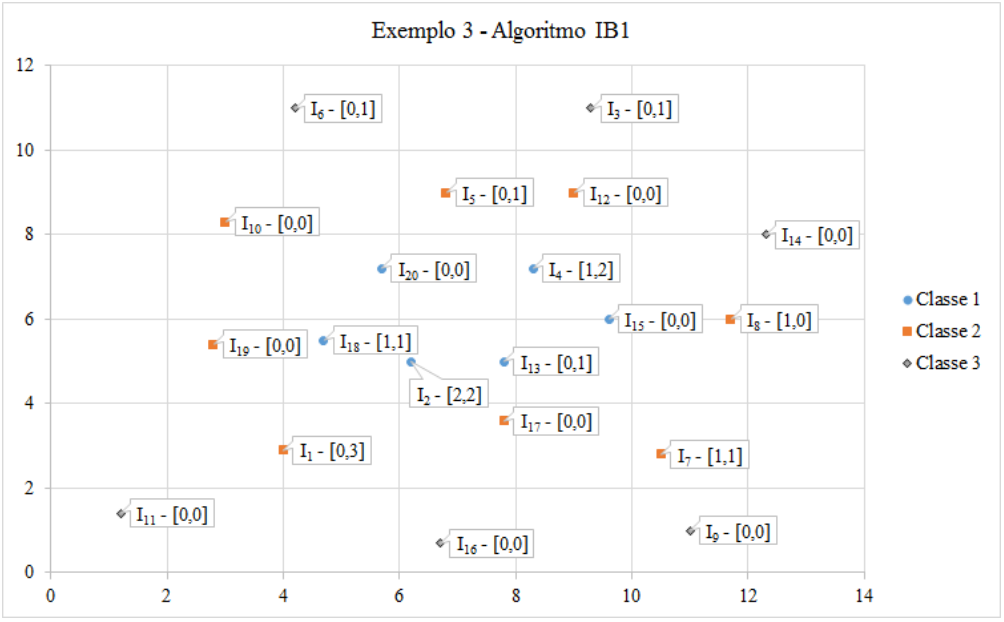


Figura 4.6 Descrição do conceito induzida pelo IB1 usando como entrada as instâncias de dados da Tabela 4.6. Cada instância é apresentada juntamente com seu registro de desempenho.

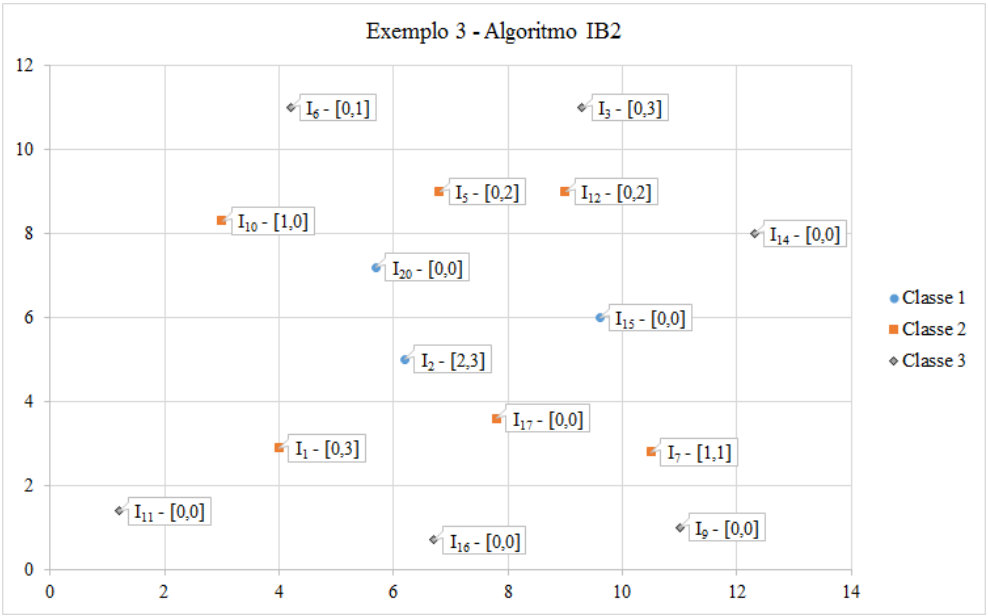


Figura 4.7 Descrição do conceito induzida pelo IB2 usando como entrada as instâncias de dados da Tabela 4.6. Cada instância é apresentada juntamente com seu registro de desempenho.

4.6 O Uso do IB2 – Comparando Resultados

Em [Aha *et al.* 1991] foram apresentados as descrições e resultados de experimentos envolvendo vários conjuntos de instâncias, com vistas à comparação entre o IB1 e IB2, com foco em precisão *versus* volume de armazenamento. A Tabela 4.11 descreve os seis domínios de dados utilizados nos experimentos (que foram extraídos do UCI ML Repository [Lichman 2013]). Tanto o número de instâncias de treinamento quanto o número de instâncias de teste usados nos experimentos descritos na referência estão reapresentados na Tabela 4.12, acompanhados dos respectivos percentuais com relação ao número total de instâncias. Nesta seção serão apresentados os resultados obtidos com o IB2-SABI, com vistas à comparação com os resultados disponibilizados em [Aha *et al.* 1991]. Para os experimentos, as instâncias de dados com valores ausentes de atributos (ver Tabela 4.11) foram previamente tratadas, por meio da imputação do valor ausente, como discutido na Seção 2.5.2.

Tabela 4.11 Características de seis domínios de dados utilizados como conjuntos de treinamento nos experimentos de comparação entre o IB1 e IB2 descritos em [Aha *et al.* 1991], em que: DD: Domínio dos dados; #TIO: Número total de instâncias no conjunto original; #ICTr: Número de instâncias no conjunto de treinamento; #ICTe: Número de instâncias no conjunto de teste; #AT: número de atributos que descrevem as instâncias; TA: Tipo dos atributo; AA/NVA: existência (ou não) de valores ausentes de atributos/número de valores ausentes; #C: Número de classes existentes no conjunto original.

DD	#TIO	#ICTr	#ICTe	#AT	TA	AA/NVA	#C
Voting	435	350	85	16	Todos Categóricos (y/n)	S/392	2
Prim.Tumor	339	275	64	17	Todos Categóricos	S/225	22
LED Display	700	200	500	7	Todos Categóricos (0/1)	N	10
Waveform	800	300	500	21	Real	N	3
Cleveland	303	250	53	13	Categóricos/Inteiro/Real	S/6	2
Hungarian	294	250	44	13	Categóricos/Inteiro/Real	S/782	2

Tabela 4.12 DD: domínio de dados; #TIO: Número total de instâncias no conjunto original; #ICTr: Número de instâncias no conjunto de treinamento (% #TIO); #ICTe: Número de instâncias no conjunto de teste (%#TIO).

DD	#TIO	#ICTr	#ICTe
Voting	435	350 (80,46%)	85 (19,54%)
Prim. Tumor	339	275 (81,12%)	64 (18,88%)
LED Display	700	200 (28,57%)	500 (71,43%)
Waveform	800	300 (37,50%)	500 (62,50%)
Cleveland	303	250 (82,51%)	53 (17,49%)
Hungarian	294	250 (85,03%)	44 (14,97%)

Com vistas a uma comparação entre os resultados reportados em [Aha *et al.* 1991] e aqueles obtidos com o uso do SABI, esta seção está organizada em sete subseções, em que a próxima descreve a metodologia empregada na condução dos experimentos com o SABI e as seis que a seguem, abordam, cada uma delas, experimentos realizados com

dados de cada um dos seis domínios descritos nas tabelas 4.11 e 4.12, na ordem em que são mostrados em tais tabelas. A última seção compara e analisa os resultados obtidos e apresentados em [Aha *et al.* 1991] e aqueles obtidos pelo SABI.

4.6.1 Descrição da Metodologia Utilizada nos Experimentos Descritos Nas Próximas Sete Subseções

A metodologia utilizada nos experimentos busca simular os experimentos descritos em [Aha *et al.* 1991]. O procedimento metodológico adotado foi: a partir do conjunto original de instâncias, com #TIO instâncias (ver Tabela 4.11), foram gerados 50 pares de conjuntos (Treinamento-Teste), contendo os percentuais de instâncias descritos nas tabelas 4.11 e 4.12.

O sistema SABI inicialmente recebe o conjunto original (CO) de instâncias e embaralha-o, de forma randômica, gerando um conjunto original embaralhado (COE). A seguir, a partir do COE são gerados 50 pares de conjuntos $Treinamento_i$–$Teste_i$ (Tr_i–Te_i), (i = 1, ..., 50), tal que $Tr_i \cap Te_i = \varnothing$ e $Tr_i \cup Te_i = COE$. Os números de instâncias de treinamento e de teste foram mantidos como aqueles do experimento descrito em [Aha *et al.* 1991].

Cada um dos conjuntos de treinamento, foi, então, utilizado como entrada para o algoritmo IB2. Ao final da fase de aprendizado, é gerado um relatório que informa o número de instâncias que foram armazenadas como parte da descrição do conceito e o número de instâncias que foram descartadas. A expressão do conceito induzida pelo IB2, usando o conjunto Tr_i foi, então, avaliada, usando o correspondente conjunto de teste Te_i (i= 1, ..., 50). Ao final da fase de avaliação do conceito induzido é gerado um relatório que informa o número de instâncias do conjunto Te_i que foram classificadas corretamente e o número de instâncias que foram classificadas incorretamente. O Capitulo 6 detalha o funcionamento do SABI, bem como suas características, funcionalidades e ferramentas.

4.6.2 Informações sobre o Domínio de Dados *Voting*

O conjunto de instâncias de dados identificado como *Congressional Voting Records Data Set* (ou simplesmente *Voting*) contém os votos dos membros do congresso americano relativos a 16 assuntos (representados pelos atributos que descrevem as instâncias). Cada um dos atributos é categórico; têm por valores apenas Y(es), N(o).

Uma instância de dado contém os votos de um único indivíduo, relativos a cada

um dos 16 assuntos e a classe associada da instância é o partido ao qual tal indivíduo pertence. Para um melhor entendimento dos dados, o nome associado a cada um dos atributos é listado a seguir, em inglês, na eventualidade de possíveis futuras comparações com resultados obtidos usando esse mesmo conjunto de dados, seguindo a notação Nomes:(valores dos atributos):

1. Class Name: (democrat, republican)
2. handicapped-infants: (y,n)
3. water-project-cost-sharing: (y,n)
4. adoption-of-the-budget-resolution: (y,n)
5. physician-fee-freeze: (y,n)
6. el-salvador-aid: (y,n)
7. religious-groups-in-schools: (y,n)
8. anti-satellite-test-ban: (y,n)
9. aid-to-nicaraguan-contras: (y,n)
10. mx-missile: (y,n)
11. immigration: (y,n)
12. synfuels-corporation-cutback: (y,n)
13. education-spending: (y,n)
14. superfund-right-to-sue: (y,n)
15. crime: (y,n)
16. duty-free-exports: (y,n)
17. export-administration-act-south-africa: (y,n)

A Tabela 4.13 mostra os resultados obtidos com o IB2 sendo executado sob o sistema SABI, seguindo a metodologia descrita em Seção 4.6.1.

Tabela 4.13 Conjunto de instâncias *VOTING*. #ICR: Número de Instâncias no Conjunto Reduzido gerado pelo IB2, a partir do conjunto de treinamento com 350 instâncias; #IDCO: Número de Instâncias Descartadas do Conjunto de Treinamento; #ITCC: Número de Instâncias do Conjunto de Teste (85 instâncias) Classificadas Corretamente; #ITCI: Número de Instâncias do Conjunto de Teste Classificadas Incorretamente; #M/DP: Média/Desvio Padrão.

	#ICR(%)	#IDCO(%)	#ITCC(%)	#ITCI(%)
E_1	39(11,14)	311(88,86)	79(92,94)	6 (7,06)
E_2	42(12,00)	308(88,00)	73(85,88)	12(14,12)
E_3	29(8,29)	321(91,71)	73(85,88)	12(14,12)
E_4	48(13,17)	302(86,29)	70(82,35)	15(17,65)
E_5	37(10,57)	313(89,43)	78(91,76)	7(8,24)
E_6	46(13,14)	304(86,86)	78(91,76)	7(8,24)
E_7	41(11,71)	309(88,29)	79(92,94)	6(7,06)
E_8	37(10,57)	313(89,43)	77(90,59)	8(9,41)
E_9	43(12,29)	307(87,71)	76(89,41)	9(10,59)
E_{10}	44(12,57)	306(87,43)	77(90,59)	8(9,41)
E_{11}	33(9,43)	317(90,57)	73(85,88)	12(14,12)
E_{12}	45(12,86)	305(87,14)	79(92,94)	6(7,06)
E_{13}	40(11,43)	310(88,57)	80(94,12)	5(5,88)
E_{14}	46(13,14)	304(86,86)	79(92,94)	6(7,06)
E_{15}	37(10,57)	313(89,43)	78(91,76)	7(8,24)
E_{16}	48(13,71)	302(86,29)	77(90,59)	8(9,41)
E_{17}	44(12,57)	306(87,43)	73(85,88)	12(14,12)
E_{18}	38(10,86)	312(89,14)	77(90,59)	8(9,41)
E_{19}	46(13,14)	304(86,86)	73(85,88)	12(14,12)
E_{20}	44(12,57)	306(87,43)	78(91,76)	7(8,24)
E_{21}	38(10,86)	312(89,14)	75(88,24)	10(11,76)
E_{22}	31(8,86)	319(91,14)	74(87,06)	11(12,94)

E_{23}	36(10,29)	314(89,71)	77(90,59)	8(9,41)
E_{24}	43(12,29)	307(87,71)	79(92,94)	6(7,06)
E_{25}	43(12,29)	307(87,71)	73(85,88)	12(14,12)
E_{26}	39(11,14)	311(88,86)	77(90,59)	8(9,41)
E_{27}	37(10,57)	313(89,43)	76(89,41)	9(10,59)
E_{28}	44(12,57)	306(87,43)	74(87,06)	11(12,94)
E_{29}	51(14,57)	299(85,43)	77(90,59)	8(9,41)
E_{30}	51(14,57)	299(85,43)	79(92,94)	6(7,06)
E_{31}	41(11,71)	309(88,29)	77(90,59)	8(9,41)
E_{32}	41(11,71)	309(88,29)	76(89,41)	9(10,59)
E_{33}	36(10,29)	314(89,71)	76(89,41)	9(10,59)
E_{34}	36(10,29)	314(89,71)	75(88,24)	10(11,76)
E_{35}	47(13,43)	303(86,57)	79(92,94)	6(7,06)
E_{36}	34(9,71)	316(90,29)	76(89,41)	9(10,59)
E_{37}	39(11,14)	311(88,86)	72(84,71)	13(15,29)
E_{38}	41(11,71)	309(88,29)	78(91,76)	7(8,24)
E_{39}	45(12,86)	305(87,14)	77(90,59)	8(9,41)
E_{40}	42(12,00)	308(88,00)	78(91,76)	7(8,24)
E_{41}	37(10,57)	313(89,43)	74(87,06)	11(12,94)
E_{42}	43(12,29)	307(87,71)	73(85,88)	12(14,12)
E_{43}	41(11,71)	309(88,29)	74(87,06)	11(12,94)
E_{44}	38(10,86)	312(89,14)	80(94,12)	5(5,88)
E_{45}	47(13,43)	303(86,57)	80(94,12)	5(5,88)
E_{46}	37(10,57)	313(89,43)	69(81,18)	16(18,82)
E_{47}	38(10,86)	312(89,14)	72(84,71)	13(15,29)
E_{48}	39(11,14)	311(88,86)	77(90,59)	8(9,41)
E_{49}	38(10,86)	312(89,14)	75(88,24)	10(11,76)
E_{50}	42(12,00)	308(88,00)	80(94,12)	5(5,88)
#M/DP	40,84 ± 4,78	309,16 ± 4,78	76,12 ± 2,71	8,88 ± 2,71
#M/DP(%)	11,67 ± 1,37	88,33 ± 1,37	89,55 ± 3,19	10,45 ± 3,19

4.6.3 Informações sobre o Domínio de Dados *Primary Tumor*

O conjunto de instâncias de dados identificado como *Primary Tumor Data Set* foi obtido junto ao Centro Médico Universitário do Instituto de Oncologia de Ljubljana, Slovenia, por M. Zwitter e M. Soklic. Por ser um conjunto de instâncias restrito é necessário enviar um pedido para ML-Repository informando o nome do conjunto, o nome de quem está solicitando e o instituto educacional ao qual o solicitante pertence, para conseguir acesso aos dados.

As instâncias são descritas por meio de 17 atributos; a classe associada a cada uma das instâncias indica se o paciente com os valores de atributos descritos na instância tem ou não tumor primário. O atributo classe possui valores numéricos que correspondem ao índice da lista do atributo *class*, mostrado a seguir. Continua válido o comentário feito anteriormente, com relação à lista de atributos que descreve as instâncias. A Tabela 4.14 mostra os resultados obtidos com o IB2 sendo executado sob o sistema SABI, seguindo a metodologia descrita em Seção 4.6.1.

1. class: lung, head & neck, esophasus, thyroid, stomach, duoden & sm.int, colon, rectum, anus, salivary glands, pancreas, gallblader, liver, kidney, bladder, testis, prostate, ovary, corpus uteri, cervix uteri, vagina, breast
2. age: <30, 30-59, >=60
3. sex: male, female
4. histologic-type: epidermoid, adeno, anaplastic

5. degree-of-diffe: well, fairly, poorly
6. bone: yes, no
7. bone-marrow: yes, no
8. lung: yes, no
9. pleura: yes, no
10. peritoneum: yes, no
11. liver: yes, no
12. brain: yes, no
13. skin: yes, no
14. neck: yes, no
15. supraclavicular: yes, no
16. axillar: yes, no
17. mediastinum: yes, no
18. abdominal: yes, no

Tabela 4.14 Conjunto de instâncias *PRIMARY TUMOR*. #ICR: Número de Instâncias no Conjunto Reduzido gerado pelo IB2, a partir do conjunto de treinamento com 275 instâncias; #IDCO: Número de Instâncias Descartadas do Conjunto de Treinamento; #ITCC: Número de Instâncias do Conjunto de Teste (64 instâncias) Classificadas Corretamente; #ITCI: Número de Instâncias do Conjunto de Teste Classificadas Incorretamente; #M/DP: Média/Desvio Padrão.

	#ICR(%)	#IDCO(%)	#ITCC(%)	#ITCI(%)
E_1	194(70,55)	81(29,45)	22(34,38)	42(65,63)
E_2	195(70,91)	80(29,09)	21(32,81)	43(67,19)
E_3	188(68,36)	87(31,64)	15(23,44)	49(76,56)
E_4	190(69,09)	85(30,91)	21(32,81)	43(67,19)
E_5	180(65,45)	95(34,55)	17(26,56)	47(73,44)
E_6	192(69,82)	83(30,18)	26(40,63)	38(59,38)
E_7	195(70,91)	80(29,09)	17(26,56)	47(73,44)
E_8	202(73,45)	73(26,55)	19(29,69)	45(70,31)
E_9	180(65,45)	95(34,55)	24(37,50)	40(62,50)
E_{10}	194(70,55)	81(29,45)	19(29,69)	45(70,31)
E_{11}	194(70,55)	81(29,45)	20(31,25)	44(68,75)
E_{12}	198(72,00)	77(28,00)	19(29,69)	45(70,31)
E_{13}	189(68,73)	86(31,27)	25(39,06)	39(60,94)
E_{14}	185(67,27)	90(32,73)	24(37,50)	40(62,50)
E_{15}	182(66,18)	93(33,82)	16(25,00)	48(75,00)
E_{16}	191(69,45)	84(30,55)	23(35,94)	41(64,06)
E_{17}	198(72,00)	77(28,00)	21(32,81)	43(67,19)
E_{18}	182(66,18)	93(33,82)	17(26,56)	47(73,44)
E_{19}	182(66,18)	93(33,82)	18(28,13)	46(71,88)
E_{20}	200(72,73)	75(27,27)	20(31,25)	44(68,75)
E_{21}	200(72,73)	75(27,27)	26(40,63)	38(59,38)
E_{22}	191(69,45)	84(30,55)	20(31,25)	44(68,75)
E_{23}	192(69,82)	83(30,18)	17(26,56)	47(73,44)
E_{24}	200(72,73)	75(27,27)	18(28,13)	46(71,88)
E_{25}	195(70,91)	80(29,09)	14(21,88)	50(78,13)
E_{26}	189(68,73)	86(31,27)	22(34,38)	42(65,63)
E_{27}	193(70,18)	82(29,82)	21(32,81)	43(67,19)
E_{28}	190(69,09)	85(30,91)	16(25,00)	48(75,00)
E_{29}	198(72,00)	77(28,00)	27(42,19)	37(57,81)
E_{30}	189(68,73)	86(31,27)	21(32,81)	43(67,19)
E_{31}	193(70,18)	82(29,82)	21(32,81)	43(67,19)
E_{32}	210(76,36)	65(23,64)	27(42,19)	37(57,81)
E_{33}	195(70,91)	80(29,09)	18(28,13)	46(71,88)
E_{34}	201(73,09)	74(26,91)	24(37,50)	40(62,50)
E_{35}	194(70,55)	81(29,45)	18(28,13)	46(71,88)
E_{36}	204(74,18)	71(25,82)	23(35,94)	41(64,06)
E_{37}	204(74,18)	71(25,82)	20(31,25)	44(68,75)
E_{38}	197(71,64)	78(28,36)	16(25,00)	48(75,00)
E_{39}	188(68,36)	87(31,64)	14(21,88)	50(78,13)
E_{40}	198(72,00)	77(28,00)	24(37,50)	40(62,50)
E_{41}	192(69,82)	83(30,18)	19(29,69)	45(70,31)
E_{42}	196(71,27)	79(28,73)	18(28,13)	46(71,88)
E_{43}	194(70,55)	81(29,45)	24(37,50)	40(62,50)
E_{44}	189(68,73)	86(31,27)	18(28,13)	46(71,88)
E_{45}	195(70,91)	80(29,09)	20(31,25)	44(68,75)

E_{46}	201(73,09)	74(26,91)	16(25,00)	48(75,00)
E_{47}	193(70,18)	82(29,82)	19(29,69)	45(70,31)
E_{48}	183(66,55)	92(33,45)	18(28,13)	46(71,88)
E_{49}	198(72,00)	77(28,00)	24(37,50)	40(62,50)
E_{50}	197(71,64)	78(28,26)	19(29,69)	45(70,31)
#M/DP	193,40 ± 6,51	81,60 ± 6,51	20,12 ± 3,36	43,88 ± 3,36
#M/DP(%)	70,33 ± 2,37	29,67 ± 2,37	31,44 ± 5,25	68,56 ± 5,25

4.6.4 Informações sobre o Domínio de Dados *LED-Display*

O conjunto de instâncias de dados identificado como *LED-Display Domain* (ou simplesmente *LED-Display*) foi construído artificialmente Aha [Aha 1988], por meio de um algoritmo programado na linguagem C.

Este conjunto de instâncias possuiu 7 atributos booleanos (0,1) que possuem ruídos com 10% de probabilidade de ter sua classe invertida; não existe nenhum valor ausente. O atributo classe possui valores numéricos que vão de 0 a 10 que representam um display de LED. Por ser um conjunto artificial os nomes dos atributos seguem uma sequência [v_1-v_7]. A Tabela 4.15 mostra os resultados obtidos com o IB2 sendo executado sob o sistema SABI, seguindo a metodologia descrita em Seção 4.6.1.

Tabela 4.15 Conjunto de instâncias *LED-DISPLAY.* #ICR: Número de Instâncias no Conjunto Reduzido gerado pelo IB2, a partir do conjunto de treinamento com 200 instâncias; #IDCO: Número de Instâncias Descartadas do Conjunto de Treinamento; #ITCC: Número de Instâncias do Conjunto de Teste (500 instâncias) Classificadas Corretamente; #ITCI: Número de Instâncias do Conjunto de Teste Classificadas Incorretamente; #M/DP: Média/Desvio Padrão.

	#ICR(%)	#IDCO(%)	#ITCC(%)	#ITCI(%)
E_1	85(42,50)	115(57,50)	292(58,40)	208(41,60)
E_2	100(50,00)	100(50,00)	261(52,20)	239(47,80)
E_3	87(43,50)	113(56,60)	329(65,80)	171(34,20)
E_4	75(37,50)	125(62,50)	301(60,20)	199(39,80)
E_5	87(43,50)	113(56,50)	295(59,00)	205(41,00)
E_6	81(40,50)	119(59,50)	330(66,00)	170(34,00)
E_7	106(53,00)	94(47,00)	269(53,80)	231(46,20)
E_8	100(50,00)	100(50,00)	254(50,80)	246(49,20)
E_9	68(34,00)	132(66,00)	353(70,60)	147(29,40)
E_{10}	94(47,00)	106(53,00)	280(56,00)	220(44,00)
E_{11}	114(57,00)	86(43,00)	242(48,40)	258(51,60)
E_{12}	76(38,00)	124(62,00)	289(57,80)	211(42,20)
E_{13}	89(44,50)	111(55,00)	322(64,40)	178(35,60)
E_{14}	72(36,00)	128(64,00)	333(66,60)	167(33,40)
E_{15}	95(47,50)	105(52,50)	312(62,40)	188(37,60)
E_{16}	95(47,50)	105(52,50)	277(55,40)	223(44,60)
E_{17}	77(38,50)	123(61,50)	345(69,00)	155(31,00)
E_{18}	70(35,00)	130(65,00)	311(62,20)	189(37,80)
E_{19}	83(41,50)	117(58,50)	323(64,60)	177(35,40)
E_{20}	85(42,50)	115(57,50)	280(56,00)	220(44,00)
E_{21}	98(49,00)	102(51,00)	294(58,80)	206(41,20)
E_{22}	60(30,00)	140(70,00)	350(70,00)	150(30,00)
E_{23}	85(42,50)	115(57,50)	285(57,00)	215(43,00)
E_{24}	71(35,50)	129(64,50)	336(67,20)	164(32,80)
E_{25}	88(44,00)	112(56,00)	335(67,00)	165(33,00)
E_{26}	90(45,00)	110(55,00)	324(64,80)	176(35,20)
E_{27}	79(39,50)	121(60,50)	353(70,60)	147(29,40)
E_{28}	56(28,00)	144(72,00)	357(71,40)	143(28,60)
E_{29}	66(33,00)	134(67,00)	366(73,20)	134(26,80)
E_{30}	83(41,50)	117(58,50)	298(59,60)	202(40,40)
E_{31}	84(42,00)	116(58,00)	257(51,40)	243(48,60)
E_{32}	88(44,00)	112(56,00)	302(60,40)	198(39,60)

E_{33}	87(43,50)	113(56,50)	285(57,00)	215(43,00)
E_{34}	93(46,50)	107(53,50)	323(64,60)	177(35,40)
E_{35}	89(44,50)	111(55,50)	313(62,60)	187(37,40)
E_{36}	91(45,50)	109(54,50)	278(55,60)	222(44,40)
E_{37}	98(49,00)	102(51,00)	273(54,60)	227(45,40)
E_{38}	70(35,00)	130(65,00)	327(65,40)	173(34,60)
E_{39}	57(28,50)	143(71,50)	368(73,60)	132(26,40)
E_{40}	75(37,50)	125(62,50)	316(63,20)	184(36,80)
E_{41}	53(26,50)	147(73,50)	361(72,20)	139(27,80)
E_{42}	106(53,00)	94(47,00)	232(46,40)	268(53,60)
E_{43}	84(42,00)	116(58,00)	266(53,20)	234(46,80)
E_{44}	95(47,50)	105(52,50)	288(57,60)	212(42,40)
E_{45}	82(41,00)	118(59,00)	268(53,60)	232(46,40)
E_{46}	92(46,00)	108(54,00)	260(52,00)	240(48,00)
E_{47}	74(37,00)	126(63,00)	307(61,40)	193(38,60)
E_{48}	84(42,00)	116(58,00)	271(54,20)	229(45,80)
E_{49}	70(35,00)	130(65,00)	332(66,40)	168(33,60)
E_{50}	87(43,50)	113(56,50)	311(62,20)	189(37,80)
#M/DP	83,48 ± 13,19	116,52 ± 13,19	304,68 ± 34,26	195,32 ± 34,26
#M/DP(%)	41,74 ± 6,59	58,26 ± 6,59	60,94 ± 6,85	39,06 ± 6,85

4.6.5 Informações sobre o Domínio de Dados *Waveform*

O conjunto de instâncias de dados identificado como *Waveform Database Generator* (ou simplesmente *Waveform*) foi construído artificialmente usando um algoritmo implementado em C, desenvolvido por Aha [Aha 1988]. Cada instância do conjunto possui 21 atributos numéricos e todos possuem ruídos, não existe nenhum valor ausente. O atributo classe possui valores numéricos que estão divididos em 3 classes (0, 1 e 2) uniformes (33% cada). Por ser um conjunto artificial os nomes dos atributos seguem uma sequência [v_1-v_{21}]. A Tabela 4.16 mostra os resultados obtidos com o IB2 sendo executado sob o sistema SABI, seguindo a metodologia descrita em Seção 4.6.1.

Tabela 4.16 Conjunto de instâncias *WAVEFORM*. #ICR: Número de Instâncias no Conjunto Reduzido gerado pelo IB2, a partir do conjunto de treinamento com 300 instâncias; #IDCO: Número de Instâncias Descartadas do Conjunto de Treinamento; #ITCC: Número de Instâncias do Conjunto de Teste (500 instâncias) Classificadas Corretamente; #ITCI: Número de Instâncias do Conjunto de Teste Classificadas Incorretamente; #M/DP: Média/Desvio Padrão.

	#ICR(%)	#IDCO(%)	#ITCC(%)	#ITCI(%)
E_1	86(28,67)	214(71,33)	328(65,60)	172(34,40)
E_2	104(34,67)	196(65,33)	298(59,60)	202(40,40)
E_3	109(36,33)	191(63,67)	351(70,20)	149(29,80)
E_4	98(32,67)	202(67,33)	349(69,80)	151(30,20)
E_5	86(28,67)	214(71,33)	358(71,60)	142(28,40)
E_6	91(30,33)	209(69,67)	362(72,40)	138(27,60)
E_7	97(32,33)	203(67,67)	342(68,40)	158(31,60)
E_8	89(29,67)	211(70,33)	369(73,80)	131(26,20)
E_9	89(29,67)	211(70,33)	353(70,60)	147(29,40)
E_{10}	96(32,00)	204(68,00)	351(70,20)	149(29,80)
E_{11}	86(28,67)	214(71,33)	355(71,00)	145(29,00)
E_{12}	95(31,67)	205(68,33)	332(66,40)	168(33,60)
E_{13}	101(33,67)	199(66,33)	328(65,60)	172(34,40)
E_{14}	100(33,33)	200(66,67)	309(61,80)	191(38,20)
E_{15}	97(32,33)	203(67,67)	372(74,40)	128(25,60)
E_{16}	109(36,33)	191(63,67)	339(67,80)	161(32,20)
E_{17}	100(33,33)	200(66,67)	320(64,00)	180(36,00)
E_{18}	124(41,33)	176(58,67)	323(64,60)	177(35,40)
E_{19}	96(32,00)	204(68,00)	317(63,40)	183(36,60)
E_{20}	102(34,00)	198(66,00)	337(67,40)	163(32,60)
E_{21}	94(31,33)	206(68,67)	355(71,00)	145(29,00)
E_{22}	104(34,67)	196(65,33)	322(64,40)	178(35,60)

E_{23}	93(31,00)	207(69,00)	363(72,60)	137(27,40)
E_{24}	91(30,33)	209(69,67)	339(67,80)	161(32,20)
E_{25}	98(32,67)	202(67,33)	360(72,00)	140(28,00)
E_{26}	95(31,67)	205(68,33)	372(74,40)	128(25,60)
E_{27}	94(31,33)	206(68,67)	347(69,40)	153(30,60)
E_{28}	119(39,67)	181(60,33)	342(68,40)	158(31,60)
E_{29}	88(29,33)	212(70,67)	343(68,60)	157(31,40)
E_{30}	102(34,00)	198(66,00)	342(68,40)	158(31,60)
E_{31}	106(35,33)	194(64,67)	339(67,80)	161(32,20)
E_{32}	103(34,33)	197(65,67)	335(67,00)	165(33,00)
E_{33}	112(37,33)	188(62,67)	335(67,00)	165(33,00)
E_{34}	68(22,67)	232(77,33)	323(64,60)	177(35,40)
E_{35}	92(30,67)	208(69,33)	378(75,60)	122(24,40)
E_{36}	101(33,67)	199(66,33)	312(62,40)	188(37,60)
E_{37}	110(36,67)	190(63,33)	339(67,80)	161(32,20)
E_{38}	106(35,33)	194(64,67)	329(65,80)	171(34,20)
E_{39}	105(35,00)	195(65,00)	343(68,60)	157(31,40)
E_{40}	105(35,00)	195(65,00)	330(66,00)	170(34,00)
E_{41}	91(30,33)	209(69,67)	333(66,60)	167(33,40)
E_{42}	105(35,00)	195(65,00)	331(66,20)	169(33,80)
E_{43}	110(36,67)	190(63,33)	351(70,20)	149(29,80)
E_{44}	92(30,67)	208(69,33)	345(69,00)	155(31,00)
E_{45}	96(32,00)	204(68,00)	355(71,00)	145(29,00)
E_{46}	87(29,00)	213(71,00)	328(65,60)	172(34,40)
E_{47}	86(28,67)	214(71,33)	345(69,00)	155(31,00)
E_{48}	105(35,00)	195(65,00)	335(67,00)	165(33,00)
E_{49}	105(35,00)	195(65,00)	337(67,40)	163(32,60)
E_{50}	95(31,67)	205(68,33)	338(67,60)	162(32,40)
#M/DP	98,26 ± 9,63	201,74 ± 9,63	304,78 ± 16,75	159,22 ± 16,75
#M/DP(%)	32,75 ± 3,21	67,25 ± 3,21	68,16 ± 3,35	31,84 ± 3,35

4.6.6 Informações sobre o Domínio de Dados *Cleveland*

Este subconjunto de instâncias pertence ao conjunto *Heart Disease Data Set*, que contém o diagnóstico de doenças cardíacas. Tais instâncias foram coletadas em quatro locais diferentes: (1) Cleveland Clinic Foundation (cleveland.data); (2) Instituto Húngaro de Cardiologia, Budapeste; (3) V.A. Medical Center, Long Beach, CA e (4) Hospital Universitário, Zurique, Suíça (switzerland.data). Os quatros conjuntos possuem a mesma estrutura com 76 atributos, sendo que apenas 14 deles são realmente utilizados, como mostrados na sequência. A Tabela 4.17 mostra os resultados obtidos com o IB2 sendo executado sob o sistema SABI, seguindo a metodologia descrita em Seção 4.6.1.

1. #3 age (int = age in years)
2. #4 sex (1 = male; 0 = female)
3. #9 cp (1 = typical angina, 2 = atypical angina, 3 = non-anginal pain, 4 = asymptomatic)
4. #10 trestbps (int = resting blood pressure)
5. #12 chol (int = serum cholestoral in mg/dl)
6. #16 fbs (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
7. #19 restecg (0 = normal, 1 = having ST-T wave abnormality, 2 = showing probable or definite left ventricular hypertrophy by Estes' criteria)
8. #32 thalach (int = maximum heart rate achieved)
9. #38 exang (exercise induced angina (1 = yes; 0 = no))
10. #40 oldpeak (decimal = ST depression induced by exercise relative to rest)
11. #41 slope (1 = upsloping, 2 = flat, 3 = downsloping)

12. #44 ca (number of major vessels (0-3) colored by flourosopy)
13. #51 thal 3 = normal; 6 = fixed defect; 7 = reversable defect)
14. #58 (num) (the predicted attribute) (0 = < 50% diameter narrowing, 1 = > 50% diameter narrowing)

Tabela 4.17 Conjunto de instâncias *CLEVELAND*. #ICR: Número de Instâncias no Conjunto Reduzido gerado pelo IB2, a partir do conjunto de treinamento com 250 instâncias; #IDCO: Número de Instâncias Descartadas do Conjunto de Treinamento; #ITCC: Número de Instâncias do Conjunto de Teste (53 instâncias) Classificadas Corretamente; #ITCI: Número de Instâncias do Conjunto de Teste Classificadas Incorretamente; #M/DP: Média/Desvio Padrão.

	#ICR(%)	#IDCO(%)	#ITCC(%)	#ITCI(%)
E_1	101(40,40)	149(59,60)	30(56,60)	23(43,40)
E_2	115(46,00)	135(54,00)	34(64,15)	19(35,85)
E_3	116(46,40)	134(53,60)	30(56,60)	23(43,40)
E_4	108(43,20)	142(56,80)	28(52,83)	25(47,17)
E_5	107(42,80)	143(57,20)	28(52,83)	25(47,17)
E_6	111(44,40)	139(55,60)	29(54,72)	24(45,28)
E_7	115(46,00)	135(54,00)	26(49,06)	27(50,94)
E_8	123(49,20)	127(50,80)	35(66,04)	18(33,96)
E_9	126(50,40)	124(49,60)	33(62,26)	20(37,74)
E_{10}	109(43,60)	141(56,40)	28(52,83)	25(47,17)
E_{11}	111(44,40)	139(55,60)	31(58,49)	22(41,51)
E_{12}	121(48,40)	129(51,06)	30(56,60)	23(43,40)
E_{13}	111(44,40)	139(55,60)	29(54,72)	24(45,28)
E_{14}	114(45,60)	136(54,40)	35(66,04)	18(33,96)
E_{15}	112(44,80)	138(55,20)	29(54,72)	24(45,28)
E_{16}	127(50,80)	123(49,20)	30(56,60)	23(43,40)
E_{17}	108(43,20)	142(56,80)	32(60,38)	21(39,62)
E_{18}	105(42,00)	145(58,00)	20(37,74)	33(62,26)
E_{19}	117(46,80)	133(53,20)	33(62,26)	20(37,74)
E_{20}	101(40,40)	149(59,60)	34(64,15)	19(35,85)
E_{21}	128(51,20)	122(48,80)	30(56,60)	23(43,40)
E_{22}	108(43,20)	142(56,80)	29(54,72)	24(45,28)
E_{23}	117(46,80)	133(53,20)	21(39,62)	32(60,38)
E_{24}	105(42,00)	145(58,00)	36(67,92)	17(32,08)
E_{25}	103(41,20)	147(58,80)	26(49,06)	27(50,94)
E_{26}	114(45,60)	136(54,40)	28(52,83)	25(47,17)
E_{27}	100(40,00)	150(60,00)	33(62,26)	20(37,74)
E_{28}	105(42,00)	145(58,00)	25(47,17)	28(52,83)
E_{29}	103(41,20)	147(58,80)	26(49,06)	27(50,94)
E_{30}	120(48,00)	130(52,00)	33(62,26)	20(37,74)
E_{31}	122(48,80)	128(51,20)	28(52,83)	25(47,17)
E_{32}	94(37,60)	156(62,40)	27(50,94)	26(49,06)
E_{33}	110(44,00)	140(56,00)	30(56,60)	23(43,40)
E_{34}	117(46,80)	133(53,20)	32(60,38)	21(39,62)
E_{35}	120(48,00)	130(52,00)	25(47,17)	28(52,83)
E_{36}	106(42,40)	144(57,60)	30(56,60)	23(43,40)
E_{37}	107(42,80)	143(57,20)	32(60,38)	21(39,62)
E_{38}	106(42,40)	144(57,60)	30(56,60)	23(43,40)
E_{39}	116(46,40)	134(53,60)	30(56,60)	23(43,40)
E_{40}	99(39,60)	151(60,40)	35(66,04)	18(33,96)
E_{41}	122(48,80)	128(51,20)	38(71,70)	15(28,30)
E_{42}	113(45,20)	137(54,80)	32(60,38)	21(39,62)
E_{43}	121(48,40)	129(51,60)	25(47,17)	28(52,83)
E_{44}	113(45,20)	137(54,80)	29(54,72)	24(45,28)
E_{45}	114(45,60)	136(54,40)	29(54,72)	24(45,28)
E_{46}	107(42,80)	143(57,20)	36(67,92)	17(32,08)
E_{47}	101(40,40)	149(59,60)	28(52,83)	25(47,17)
E_{48}	104(41,60)	146(58,40)	28(52,83)	25(47,17)
E_{49}	99(39,60)	151(60,40)	26(49,06)	27(50,94)
E_{50}	109(43,60)	141(56,40)	29(54,72)	24(45,28)
#M/DP	111,22 ± 8,03	138,78 ± 8,03	29,80 ± 3,66	23,30 ± 3,66
#M/DP(%)	44,49 ± 3,21	55,51 ± 3,21	56,23 ± 6,91	43,77 ± 6,91

4.6.7 Informações sobre o Domínio de Dados *Hungarian*

Este subconjunto de instâncias é parte do conjunto *Heart Disease Data Set*, discutido e apresentado na Seção 4.6.6. A Tabela 4.18 mostra os resultados obtidos com o IB2 sendo executado sob o sistema SABI, seguindo a metodologia descrita em Seção 4.6.1.

Tabela 4.18 Conjunto de instâncias *HUNGARIAN*. #ICR: Número de Instâncias no Conjunto Reduzido gerado pelo IB2, a partir do conjunto de treinamento com 250 instâncias; #IDCO: Número de Instâncias Descartadas do Conjunto de Treinamento; #ITCC: Número de Instâncias do Conjunto de Teste (44 instâncias) Classificadas Corretamente; #ITCI: Número de Instâncias do Conjunto de Teste Classificadas Incorretamente; #M/DP: Média/Desvio Padrão.

	#ICR(%)	#IDCO(%)	#ITCC(%)	#ITCI(%)
E_1	106(42,40)	144(57,60)	22(50,00)	22(50,00)
E_2	112(44,80)	138(55,20)	25(56,82)	19(43,18)
E_3	109(43,60)	141(56,40)	23(52,27)	21(47,73)
E_4	96(38,40)	154(61,60)	25(56,82)	19(43,18)
E_5	112(44,80)	138(55,20)	26(59,09)	18(40,91)
E_6	95(38,00)	155(62,00)	29(65,91)	15(34,09)
E_7	115(46,00)	135(54,00)	25(56,82)	19(43,18)
E_8	118(47,20)	132(52,80)	29(65,91)	15(34,09)
E_9	112(44,80)	138(55,20)	25(56,82)	19(43,18)
E_{10}	106(42,40)	144(57,60)	27(61,36)	17(38,64)
E_{11}	100(40,00)	150(60,00)	24(54,55)	20(45,45)
E_{12}	107(42,80)	143(57,20)	27(61,36)	17(38,64)
E_{13}	111(44,40)	139(55,60)	22(50,00)	22(50,00)
E_{14}	116(46,40)	134(53,60)	28(63,64)	16(36,36)
E_{15}	106(42,40)	144(57,60)	23(52,27)	21(47,73)
E_{16}	101(40,40)	149(59,60)	22(50,00)	22(50,00)
E_{17}	115(46,00)	135(54,00)	25(56,82)	19(43,18)
E_{18}	112(44,80)	138(55,20)	23(52,27)	21(47,73)
E_{19}	99(39,60)	151(60,40)	25(56,82)	19(43,18)
E_{20}	108(43,20)	142(56,80)	28(63,64)	16(36,36)
E_{21}	112(44,80)	138(55,20)	29(65,91)	15(34,09)
E_{22}	111(44,40)	139(55,60)	26(59,09)	18(40,91)
E_{23}	111(44,40)	139(55,60)	25(56,82)	19(43,18)
E_{24}	107(42,80)	143(57,20)	27(61,36)	17(38,64)
E_{25}	109(43,60)	141(56,40)	24(54,55)	20(45,45)
E_{26}	99(39,60)	151(60,40)	25(56,82)	19(43,18)
E_{27}	108(43,20)	142(56,80)	25(56,82)	19(43,18)
E_{28}	112(44,80)	138(55,20)	24(54,55)	20(45,45)
E_{29}	114(45,60)	136(54,40)	24(54,55)	20(45,45)
E_{30}	118(47,20)	132(52,80)	22(50,00)	22(50,00)
E_{31}	107(42,80)	143(57,20)	23(52,27)	21(47,73)
E_{32}	114(45,60)	136(54,40)	25(56,82)	19(43,18)
E_{33}	105(42,00)	145(58,00)	29(65,91)	15(34,09)
E_{34}	110(44,00)	140(56,00)	21(47,73)	23(52,27)
E_{35}	108(43,20)	142(56,80)	31(70,45)	13(29,55)
E_{36}	108(43,20)	142(56,80)	28(63,64)	16(36,36)
E_{37}	110(44,00)	140(56,00)	24(54,55)	20(45,45)
E_{38}	106(42,40)	144(57,60)	23(52,27)	21(47,73)
E_{39}	111(44,40)	139(55,60)	21(47,73)	23(52,27)
E_{40}	117(46,80)	133(53,20)	26(59,09)	18(40,91)
E_{41}	114(45,60)	136(54,40)	25(56,82)	19(43,18)
E_{42}	113(45,20)	137(54,80)	27(61,36)	17(38,64)
E_{43}	116(46,40)	134(53,60)	22(50,00)	22(50,00)
E_{44}	104(41,60)	146(58,40)	25(56,82)	19(43,18)
E_{45}	104(41,60)	146(58,40)	23(52,27)	21(47,73)
E_{46}	113(45,20)	137(54,80)	24(54,55)	20(45,45)
E_{47}	115(46,00)	135(54,00)	21(47,73)	23(52,27)
E_{48}	108(43,20)	142(56,80)	29(65,91)	15(34,09)
E_{49}	117(46,80)	133(53,20)	19(43,18)	25(56,82)
E_{50}	120(48,00)	130(52,00)	21(47,73)	23(52,27)
#M/DP	109,54 ± 5,73	140,46 ± 5,76	24,82 ± 2,62	19,18 ± 2,62
#M/DP(%)	43,82 ± 2,29	56,18 ± 2,29	56,41 ± 5,95	43,59 ± 5,95

4.6.8 Comparando os Resultados Obtidos pelo IB2 em Experimentos Realizados com Implementações Distintas

A Tabela 4.19 mostra os resultados dos experimentos descritos em [Aha *et al.*1991], para cada um dos seis domínios de dados utilizados. Nesta seção é de interesse apenas a coluna IB2 da Tabela 4.19, que será comparada aos resultados obtidos pelo IB2 implementado sob o sistema computacional SABI, como mostrados na Tabela 4.20. protótipo

Tabela 4.19 Resultados dos experimentos descritos em [Aha *et al.* 1991], utilizando o IB1 e o IB2, nos seis domínios de dados das tabelas 4.11 e 4.12, mostrando a Precisão ± desvio padrão(DP) e o percentual de armazenamento (% AR). Na tabela o ponto, usado como separador entre a parte inteira e a parte decimal foi mantido, como está no original.

Domínios de dados	IB1		IB2	
	Precisão±DP	% AR	Precisão±DP	% AR
Voting	91.4 ± 0.4	100	90.9 ± 0.5	11.1
Primary Tumor	34.7 ± 0.8	100	32.9 ± 0.8	71.3
LED Display	70.5 ± 0.4	100	62.4 ± 0.6	41.5
Waveform	75.2 ± 0.3	100	69.6 ± 0.4	32.5
Cleveland	75.7 ± 0.8	100	71.4 ± 0.8	30.4
Hungarian	58.7 ± 1.5	100	55.9 ± 2.0	36.0

Tabela 4.20 Resultados dos experimentos realizados neste trabalho, utilizando a implementação do IB2 disponibilizada sob o sistema computacional SABI. Para cada um dos seis domínios de são mostrados os valores de Precisão ± desvio padrão(DP) e o percentual de armazenamento (% AR).

Domínios de dados	IB2 (SABI)	
	Precisão±DP	% AR
Voting	85.55 ± 3.19	11.67
Primary Tumor	68.56 ± 5.25	70.33
LED Display	60.94 ± 6.85	41.74
Waveform	68.16 ± 3.35	32.75
Cleveland	56.23 ± 6.91	44.49
Hungarian	56.41 ± 5.95	43.82

Os resultados do sistema SABI, apresentaram consistência aos obtidos no experimento descrito em [Aha *et al.*1991], em precisão e armazenamento, as diferenças apresentadas podem ser justificada pelo desconhecimento da ordem em que as instâncias foram fornecidas ao algoritmo IB2.

Capítulo 5
Algoritmos CNN, RNN E METHOD$_2$

5.1 Considerações Iniciais

As próximas cinco seções que compõem esse capítulo têm foco em algoritmos que buscam minimizar o volume de instâncias de dados a serem armazenadas como expressão do conceito, quando do uso do NN (*Nearest Neighbour*), abordado na Seção 4.4. Com a introdução de uma estratégia para minimizar o volume de armazenamento, tais algoritmos são considerados versões do NN que foram modificadas de maneira a agregarem estratégias de minimização do volume de instâncias armazenadas.

5.2 O Algoritmo CNN (*Condensed Nearest Neighbour*)

Considere novamente o algoritmo NN como proposto em [Cover & Hart 1967] e apresentado em Algoritmo 2.1, no Capítulo 2. Descrito de uma maneira simples, o NN armazena o conjunto de treinamento todo (T_{NN}) e, quando da classificação de uma nova instância, classifica-a com a classe da instância armazenada que lhe for mais próxima.

A variante do NN, conhecida como CNN (*Condensed Nearest Neighbour*), proposta em [Hart 1968], procura reduzir o volume de dados armazenados pelo NN (*i.e.*, o conjunto notado por T_{NN} do Algoritmo 2.1), por meio da escolha de um subconjunto representativo dos dados em T_{NN}, notado por T_{CNN}. Como apontado por Hart em [Hart 1968], o NN não é o algoritmo escolhido em muitas aplicações devido às exigências de armazenamento que impõe.

O CNN retém a abordagem básica do NN sem, no entanto, impor as exigências de armazenamento, como pode ser visto no Algoritmo 5.1, em uma reescrita do algoritmo descrito textualmente em [Hart 1968], mantendo a nomenclatura original relativa à duas das variáveis do algoritmo original, STORE e GRABBAG. A opção por manter os dois nomes de variáveis foi motivada para facilitar uma comparação dos dois pseudocódigos, aquele publicado em [Hart 1968] e o produzido neste trabalho. Também, nos algoritmos, tanto STORE quanto GRABBAG são tratadas como conjuntos em que a ordem em que os elementos aparecem no conjunto original é mantida.

```
procedure cnn(T_NN)

Input: conjunto de instâncias T_NN
Output: conjunto de instâncias STORE (T_CNN)

GRABBAG ← ∅
STORE ← primeira instância de T_NN
I ← primeira instância de T_NN
while percorre(T_NN) do
begin
    if classifica_correto(I,STORE)
      then begin
             GRABBAG ← GRABBAG ∪ {I}
             I ← next_instance(T_NN)
           end
      else begin
             STORE ← STORE ∪ {I}
             I ← next_instance(T_NN)
           end
end
if GRABBAG = ∅ then return(STORE)
% loop de varredura de GRABBAG para refinar SOURCE
loop ← true
checkbag ← true
while loop do
    begin
      I ← next_instance(GRABBAG)
      while checkbag do
       begin
         if classifica_correto(I,STORE)
                  then
                    if GRABBAG = ∅ then loop ← false
                  else
                    begin
                      STORE ← STORE ∪ {I}
                      checkbag ← false
                      GRABBAG ← GRABBAG – {I}
                      call reset_pointer_to_beginning_of_GRABBAG
                    end
       end
    end
if is_empty(GRABBAG)
     then message("subconjunto consistente é o próprio conjunto")
return STORE    % STORE = T_CNN
end procedure
```

Algoritmo 5.1 Reescrita do pseudocódigo do algoritmo CNN (*Condensed Nearest Neighbour*) com base na descrição textual do algoritmo encontrada em [Hart 1968].

Para a construção do conjunto T_{CNN} o algoritmo CNN usa o conceito de *subconjunto consistente*, definido como um subconjunto do conjunto T_{NN} que classifica todas as instâncias de dados em T_{NN}. O subconjunto minimal é o menor dos

subconjuntos consistentes e, consequentemente, aquele que irá classificar de maneira mais eficiente e apropriada, toda instância de T_{NN}. O algoritmo CNN, entretanto, encontra sempre um subconjunto consistente que, não necessariamente, é minimal. Assumindo que o conjunto inicial de instâncias T_{NN} é consistente (*i.e.*, uma mesma instância não comparece em T_{NN} associada a duas classes diferentes), o subconjunto T_{CNN} classifica todos as instâncias de T_{NN} corretamente.

O CNN, portanto, não é um novo algoritmo de classificação mas sim, é um algoritmo que, como o NN, ainda faz a classificação de uma nova instância com base na instância que lhe for mais próxima. A diferença entre os dois, portanto, é o conjunto de treinamento que ambos armazenam como expressão do conceito. Enquanto o NN armazena todo o conjunto de treinamento, o CNN busca reduzir o conjunto original e, com isso, as exigências de armazenamento. Como comentado em [Gates 1972], uma possível queda em eficiência (quando da classificação de instâncias com classes desconhecidas) do CNN é, geralmente, compensada pela maior eficiência tanto no volume de memória utilizado para armazenar o conjunto de treinamento quanto no tempo computacional necessário para realizar a classificação.

De acordo com Hart em [Hart 1968], se as densidades das várias classes representadas no conjunto de treinamento têm pouca sobreposição, a tendência do CNN é escolher instâncias perto da fronteira entre classes. Aquelas instâncias que estão completamente imersos dentro de uma classe não serão transferidos para STORE, uma vez que serão corretamente classificados. Se as classes têm uma sobreposição razoável, entretanto, a tendência é que STORE contenha praticamente todas as instâncias do conjunto original; neste caso o algoritmo não terá realizado qualquer redução perceptível no conjunto original de instâncias de dados.

5.3 Um Exemplo do Uso do CNN (*Condensed Nearest Neighbour*)

O exemplo do uso do CNN, descrito a seguir, foi extraído de [Gates 1972]. Considere o conjunto de treinamento com 13 instâncias de dados (unidimensionais), como descritas na Tabela 5.1. Cada instância é descrita pelo valor de um único atributo (X) e pela classe associada.

Tabela 5.1 Conjunto de treinamento TR com 13 instâncias de dados unidimensionais e respectiva classe, sendo 6 de classe 1 e 7 de classe 2.

#ID	X	Classe	#ID	X	Classe
I_1	−13	1	I_8	2	2
I_2	−16	1	I_9	4	2
I_3	−19	1	I_{10}	6	2
I_4	−6	2	I_{11}	13	1
I_5	−4	2	I_{12}	16	1
I_6	−2	2	I_{13}	19	1
I_7	0	2			

O primeiro elemento de TR é a instância (−13,1), cujo valor de seu único atributo é −13 e que pertence à classe 1. O conceito aprendido pelo NN (T_{NN}) é o próprio conjunto de treinamento *i.e.*, T_{NN} = {(−13,1), (−16,1), (−19,1), (−6,2), (−4,2), (−2,2), (0,2), (2,2), (4,2), (6,2), (13,1), (16,1), (19,1)}.

A Figura 5.1 mostra uma representação gráfica da expressão do conceito pelo NN (T_{NN}). A Figura 5.2 mostra um *trace* da obtenção da expressão reduzida do conceito, pelo CNN (T_{CNN}).

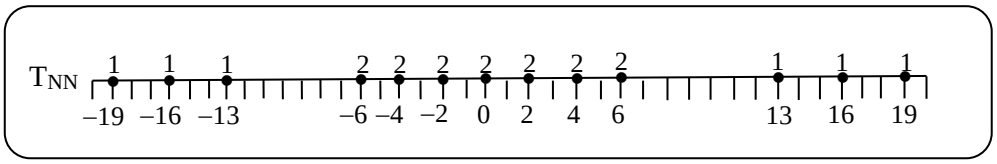


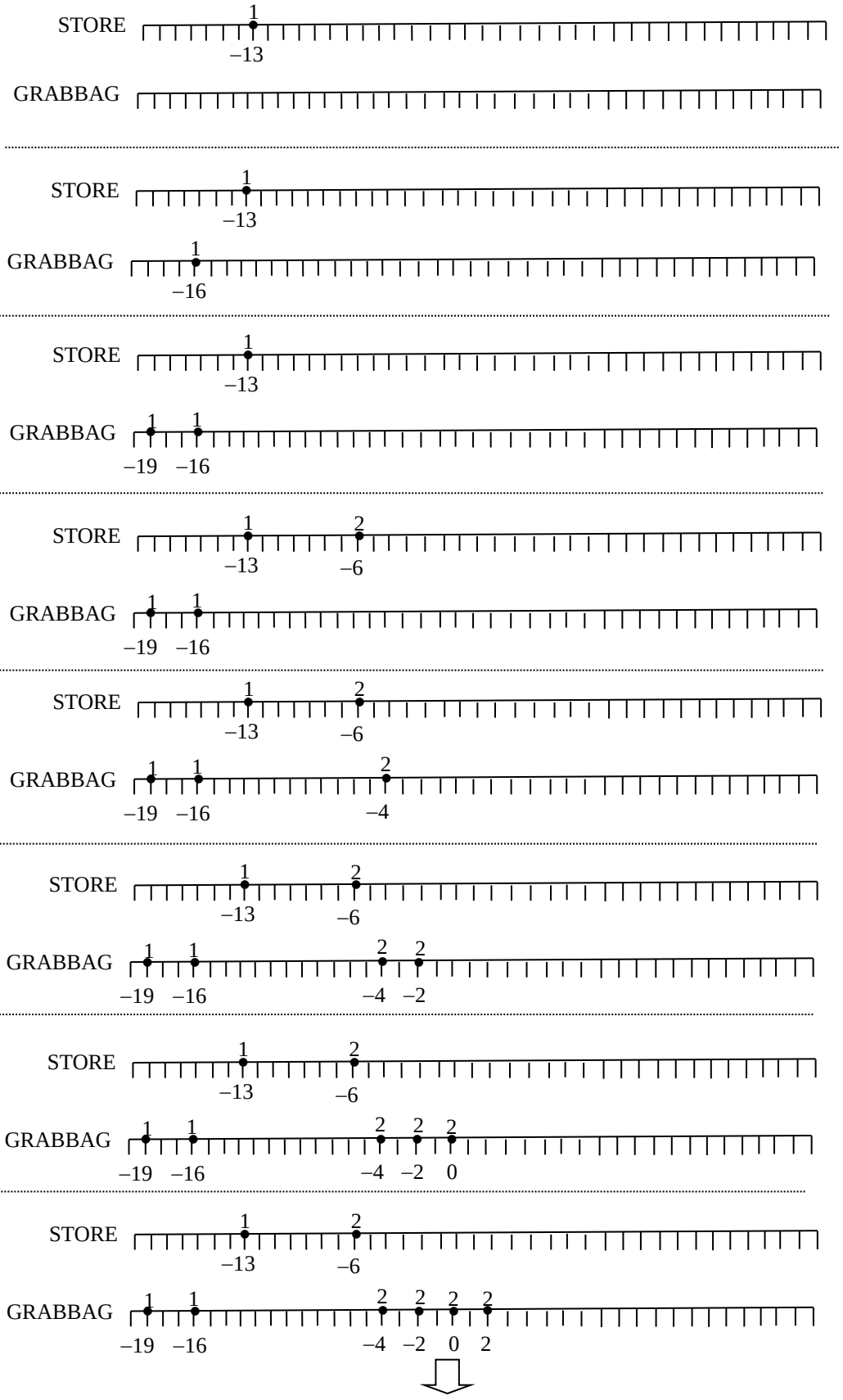
Figura 5.1 O conjunto de treinamento TR (mostrado na Tabela 5.1), com 13 instâncias e duas possíveis classes (1 e 2) é aprendido pelo NN como o conjunto T_{NN}, que é o próprio conjunto TR.

Note na parte final do *trace* mostrado na Figura 5.2 que, quanto a instância (19,1) é introduzida em STORE, a primeira varredura do conjunto de instâncias termina. O algoritmo CNN continua por meio de um *loop* de varredura, dessa vez das instâncias em GRABBAG, checando se cada uma delas está corretamente classificada pelo conjunto de instâncias em STORE. Quando alguma delas não estiver, ela é movida para STORE e uma nova varredura de GRABBAG é feita. Tal *loop* termina quando uma das duas condições seguintes acontecer:

(1) GRABBAG se torna vazia, com todos seus elementos transferidos para STORE. Quando essa situação acontece, todo o conjunto original de instâncias é o subconjunto consistente procurado, ou

(2) Uma varredura completa é feita em GRABBAG sem que aconteça qualquer transferência de qualquer de suas instâncias para STORE. Isso acontecendo, todas as varreduras subsequentes de GRABBAG irão resultar em nenhuma transferência, uma

vez que o conjunto de instâncias que classifica (i.e., aquele armazenado em STORE), não mudou.



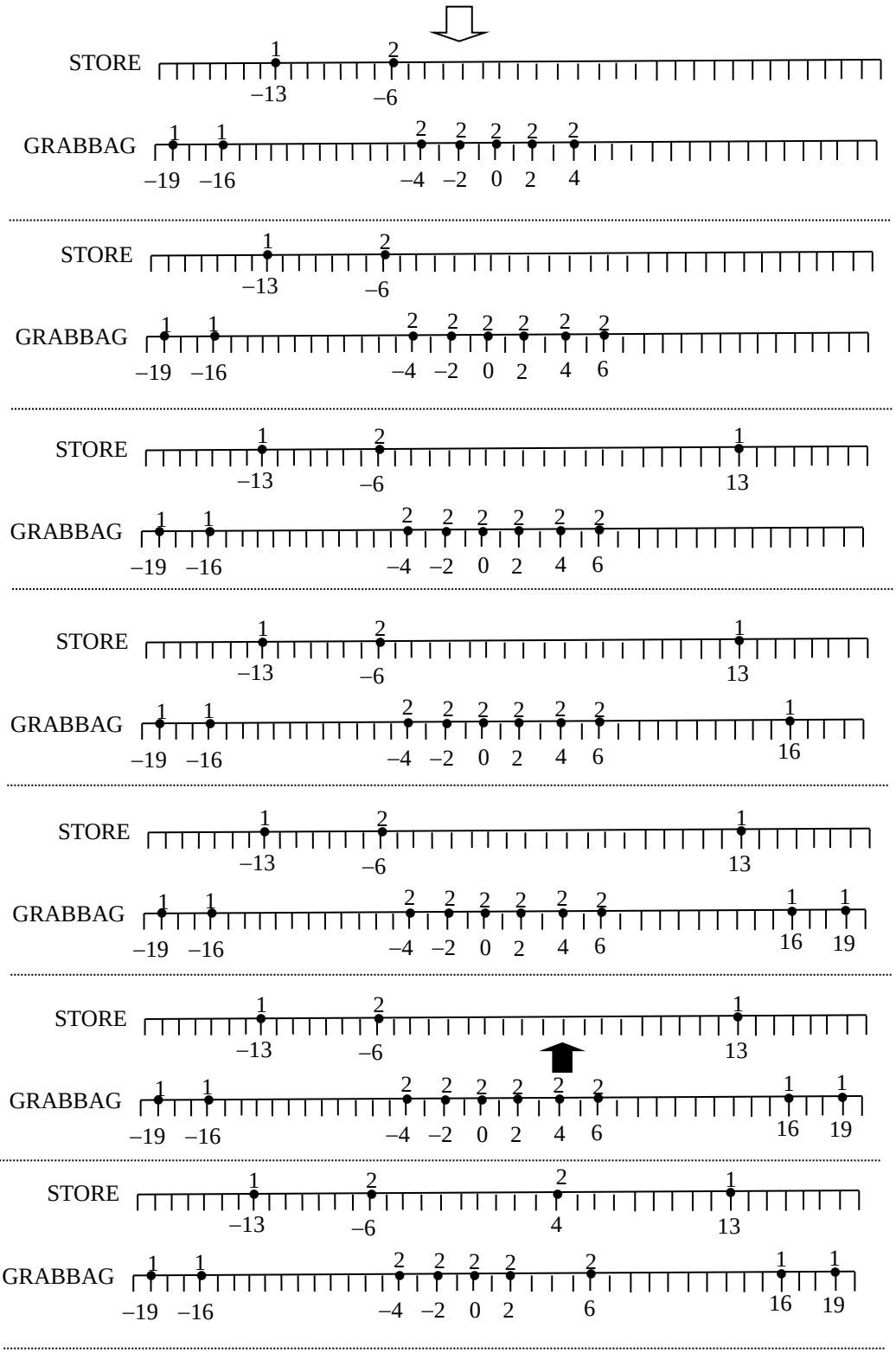


Figura 5.2 O conjunto de treinamento TR (mostrado na Tabela 5.1), com 13 instâncias e duas possíveis classes (1 e 2) é aprendido pelo C_{NN}, como o conjunto T_{CNN} = {(–13,1), (–6,2), (4,2), (13,1)}, armazenado em STORE. Note que quanto a instância (19,1) é introduzida em STORE, a primeira varredura do conjunto de instâncias termina. O CNN inicia então a varredura de GRABBAG e detecta que a instância (4,2) não é classificada corretamente na classe 2 mas, sim, na classe 1, dado que está a uma distância menor da instância (13, 1) do que da instância (–6,2). Ela é movida então para STORE e uma nova varredura se inicia. Como o restante das instâncias em GRABAG é corretamente classificado pelas instâncias em STORE, o CNN finaliza e em

STORE está armazenado um conjunto reduzido de instâncias (T_{CNN}) extraído do conjunto original, que classifica corretamente todo o conjunto original de instâncias.

5.4 O Algoritmo RNN (*Reduced Nearest Neighbour*)

O RNN (*Reduced Nearest Neighbor*) foi proposto por Gates em [Gates 1972] como uma extensão do CNN e, de maneira similar ao CNN, o RNN reduz o conjunto original de instâncias, T_{NN}. Como o RNN é inicializado considerando o resultado do CNN, Gates também apresenta uma reescrita do CNN, adaptada ao uso que tal autor faz de tal algoritmo. Tal descrição é apresentada em Algoritmo 5.2, como o procedimento *cnn_gates*. O RNN apresentado no Algoritmo 5.3 tem, como primeiro passo, a execução do CNN.

procedure *cnn_gates*

Input: conjunto de instâncias T_{NN}
Output: conjunto reduzido de instâncias T_{CNN}

1. T_{CNN}← primeira instância de T_{NN}
2. T_{CNN} é usado para classificar cada instância de T_{NN}, a partir da primeira.
   Esse processo é repetido até um dos dois casos surgir:
   (a) toda instância em T_{NN} é classificada corretamente, o que termina o processo;
   (b) uma das instâncias em T_{NN} é classificada incorretamente, quando 3. deve ser acionado.
3. Adicione a instância de T_{NN} que foi classificada incorretamente a T_{CNN} e retorne a 2.

return T_{CNN}
end procedure

Algoritmo 5.2 Algoritmo CNN (*Condensed Nearest Neighbour*) descrito por Gates em [Gates 1972].

Na referência [Gates 1972] o autor pondera que, uma vez que o RNN é uma extensão do CNN, o número de vizinhos mais próximos da nova instância a ser classificada, no RNN, será também 1. O algoritmo RNN proposto em [Gates 1972] está descrito em Algoritmo 5.3. Como comentado anteriormente, na formação do T_{CNN} a noção de *subconjunto consistente* de T_{NN} é fundamental; tal subconjunto é um que classifica todas as instâncias em T_{NN}, de maneira correta. O subconjunto consistente minimal é o menor deles (em número de elementos) e, portanto, é o subconjunto mais eficiente de T_{NN} que irá classificar apropriadamente toda instância em T_{NN}. Como lembra Gates em [Gates 1972], o *subconjunto consistente minimal* de T_{NN} é importante

uma vez que, de alguma forma, ele generaliza toda informação relevante sobre o T_{NN}. Lembrando novamente, o T_{CNN} constrói um subconjunto consistente, mas sem a garantia que seja minimal.

```
procedure rnn

Input: conjunto de instâncias T_NN                    % T_NN = {I_1, I_2,..., I_N}
Output: T_RNN

T_CNN ← cnn(T_NN)
T_RNN ← T_CNN
while existirem instâncias em T_CNN que não foram removidas uma vez do
    begin
      FP ← remove_next_instance(T_RNN)  % remoção feita sequencialmente
      okay ← classify(T_NN, T_RNN)      % classifica T_NN usando R_RNN
      if not okay then T_RNN ← {FP} ∪ T_RNN
    end
return T_RNN
end procedure
```

Algoritmo 5.3 Reescrita do algoritmo RNN (*Reduced Nearest Neighbour*) como proposto em [Gates 1972].

Com relação ao algoritmo RNN um aspecto a ser considerado diz respeito ao conjunto gerado pelo algoritmo, o T_{RNN}, ser (ou não) um subconjunto consistente minimal de T_{NN}. Obviamente, desde que $T_{RNN} \subseteq T_{CNN}$, se o T_{CNN} não contiver um subconjunto consistente minimal de T_{NN}, tampouco o T_{RNN} conterá tal subconjunto.

5.4.1 Uso do RNN em Situação em que o T_{CNN} Não Contém o Subconjunto Consistente Minimal

O exemplo tratado na Seção 5.3 exibe uma situação em que o T_{CNN} não contém um subconjunto consistente minimal de T_{NN} e, portanto, tampouco o T_{RNN} o terá.

Note que para o exemplo em questão, T_{NN} = {(–13,1), (–16,1), (–19,1), (–6,2), (–4,2), (–2,2), (0,2), (2,2), (4,2), (6,2), (13,1), (16,1), (19,1)}. Ao final da Seção 5.3 é exibido um *trace* de execução do CNN, sendo que o resultado obtido foi T_{CNN} = {(–13,1), (–6,2), (4,2), (13,1)}. Esse resultado se mantém, quando da execução do RNN que tem como input tal T_{CNN} (*i.e.*, um T_{CNN} que não contém um subconjunto consistente minimal).

No exemplo, o subconjunto consistente minimal é o $T_{minimal}$ = {(–13,1), (0,2), (13,1)}. Note que $T_{minimal}$ classifica corretamente todo o T_{NN} (como pode ser visualizado

na Figura 5.3) e que $|T_{minimal}| = 3$, enquanto que $|T_{CNN}| = |T_{RNN}| = 4$.

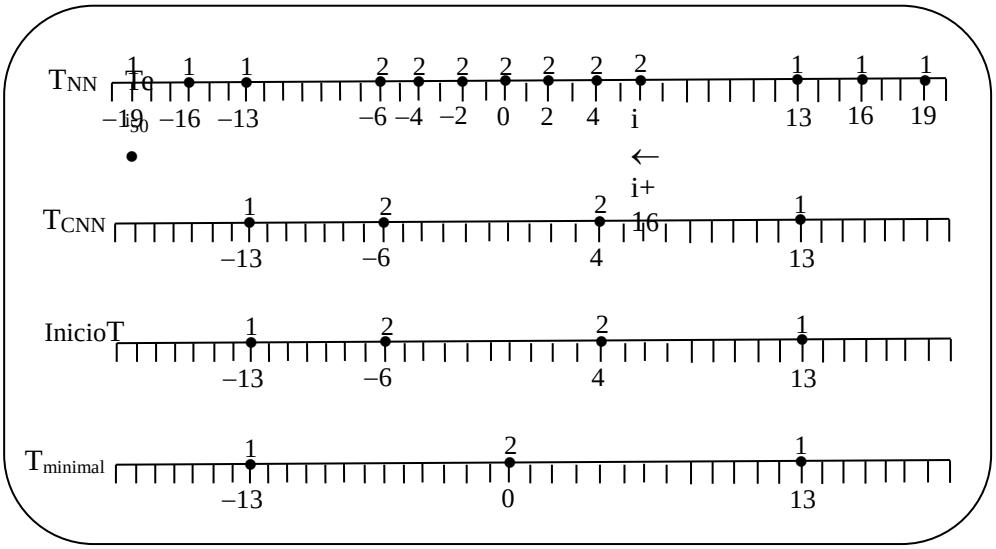


Figura 5.3 O conjunto de treinamento TR com 13 instâncias é aprendido pelo NN como o conjunto T_{NN}, pelo C_{NN}, como o conjunto T_{CNN} = {(−13,1), (−6,2), (4,2), (13,1)}. Quando o RNN é utilizado, ele mantém o conjunto aprendido pelo CNN *i.e.*, T_{RNN} = {(−13,1), (−6,2), (4,2), (13,1)}. O subconjunto consistente minimal, entretanto, é $T_{minimal}$ = {(−13,1), (0,2), (13,1)}. Note que subconjunto minimal não é subconjunto de T_{CNN} e, portanto, não pode ser aprendido pelo RNN.

5.4.2 Uso do RNN em Situação em que o T_{CNN} Contém o Conjunto Consistente Minimal

O exemplo que segue, extraído de [Gates 1972], mostra o que acontece quando o T_{CNN} tem um subconjunto consistente minimal.

Considere T_{NN} = {(0,2), (−3,1), (−2,2), (3,1), (2,2)}. Como pode ser visto no *trace* mostrado na Figura 5.4, considerando que as instâncias tenham sido fornecidas ao CNN na ordem em que se encontram no conjunto acima, todas as instâncias são incorporadas diretamente na variável STORE e a variável GRABBAG não é utilizada. Portanto, $T_{CNN} = T_{NN}$ = {(0,2), (−3,1), (−2,2), (3,1), (2,2)}.

No entanto, quando o RNN é executado, o resultado é T_{RNN} = { (−3,1), (−2,2), (3,1), (2,2)} e $T_{minimal} = T_{RNN}$. O que torna o algoritmo RNN uma ferramenta útil é o fato que sempre que $T_{minimal} \subseteq T_{CNN}$, $T_{RNN} = T_{minimal}$ (uma discussão sobre esse resultado teórico pode ser encontrada em [Gates 1972]).

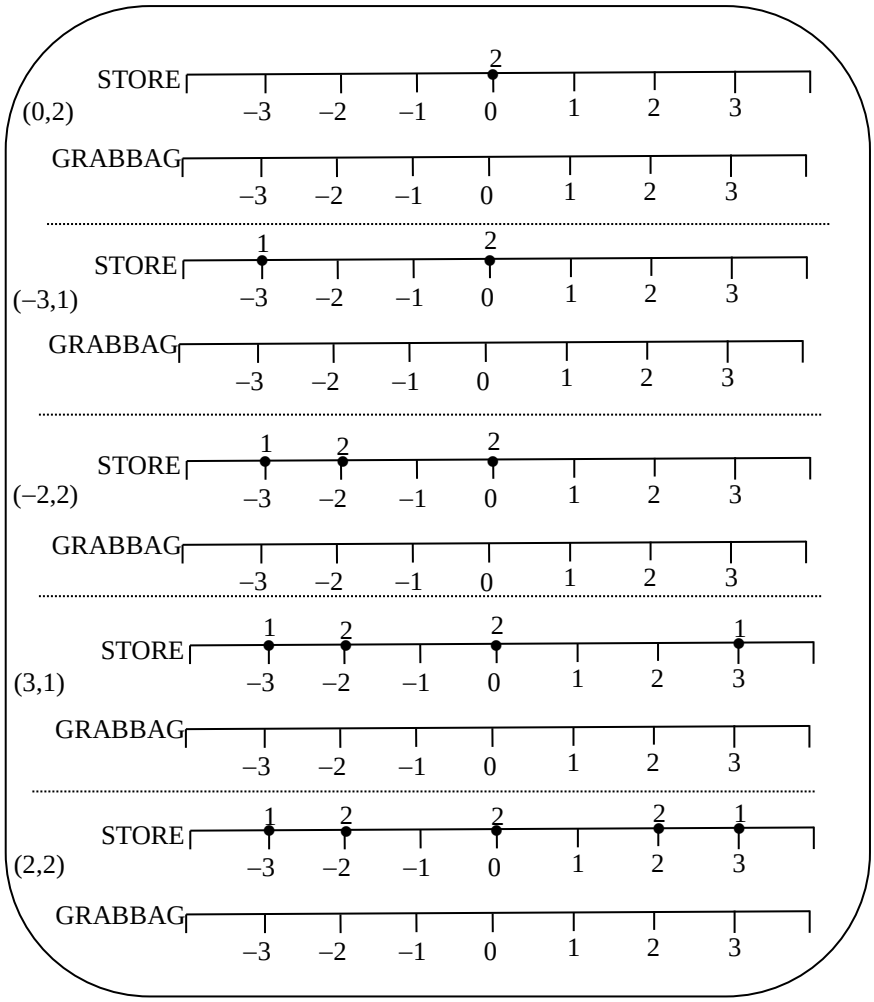


Figura 5.4 Uso do CNN no conjunto T_{NN} = {(0,2), (–3,1), (–2,2), (3,1), (2,2)}, com as instâncias processadas na ordem em que aparecem no conjunto. O conjunto original não é reduzido e $T_{CNN} = T_{NN}$ = {(0,2), (–3,1), (–2,2), (3,1), (2,2)}.

A Figura 5.5 mostra os passos do algoritmo RNN no conjunto T_{CNN} = {(0,2), (–3,1), (–2,2), (3,1), (2,2)}, com as instâncias processadas na ordem em que aparecem no conjunto.

Como mostra a Figura 5.4, o conjunto original não é reduzido e $T_{CNN} = T_{NN}$ = {(0,2), (–3,1), (–2,2), (3,1), (2,2)}. O uso do RNN tendo como ponto de partida o conjunto T_{CNN} (que, no caso, é o próprio T_{NN}), produz o conjunto $T_{RNN} = T_{minimal}$ = {(–3,1), (–2,2), (3,1), (2,2)}.

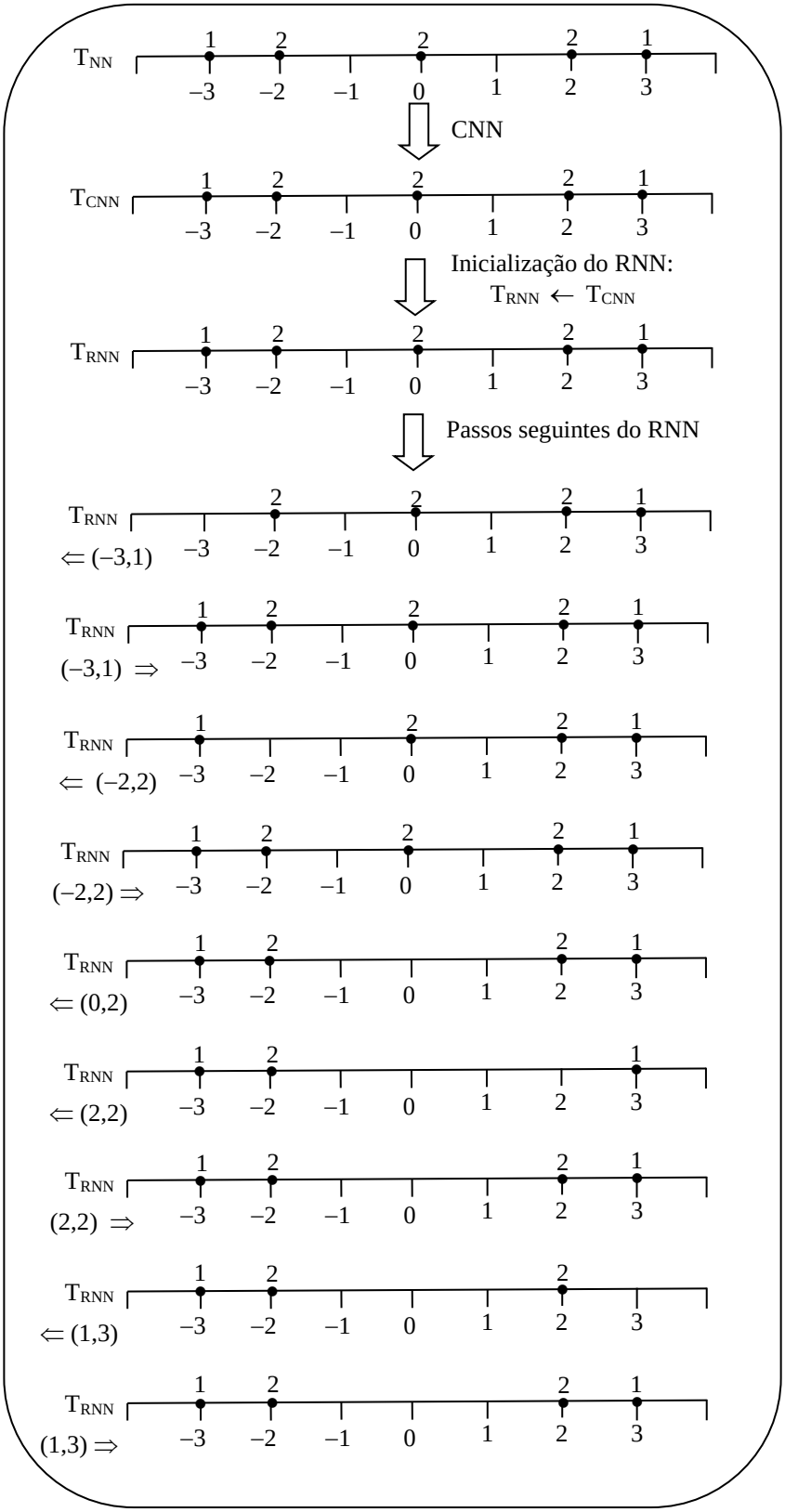


Figura 5.5 Uso do RNN no conjunto T_{CNN} = {(0,2), (–3,1), (–2,2), (3,1), (2,2)}, com as instâncias processadas na ordem em que aparecem no conjunto. Como visto na Figura 5.4, o conjunto original não é reduzido e $T_{CNN} = T_{NN}$ = {(0,2), (–3,1), (–2,2), (3,1), (2,2)}. No entanto, o uso do RNN tendo como ponto de partida o conjunto T_{CNN}, produz o conjunto $T_{RNN} = T_{minimal}$ = {(–3,1), (–2,2), (3,1), (2,2)}.

5.5 Sobre a Relevância da Ordem das Instâncias no Conjunto a ser Reduzido

Como comentado anteriormente, dado um conjunto consistente de instâncias *i.e.*, não existem no conjunto duas instâncias iguais pertencentes a classes diferentes), o CNN sempre encontra um subconjunto consistente que, não necessariamente, é minimal. Pode acontecer que, dependendo da ordem em que as instâncias são apresentadas ao algoritmo, o CNN induza o subconjunto consistente minimal como mostra a Figura 5.6.

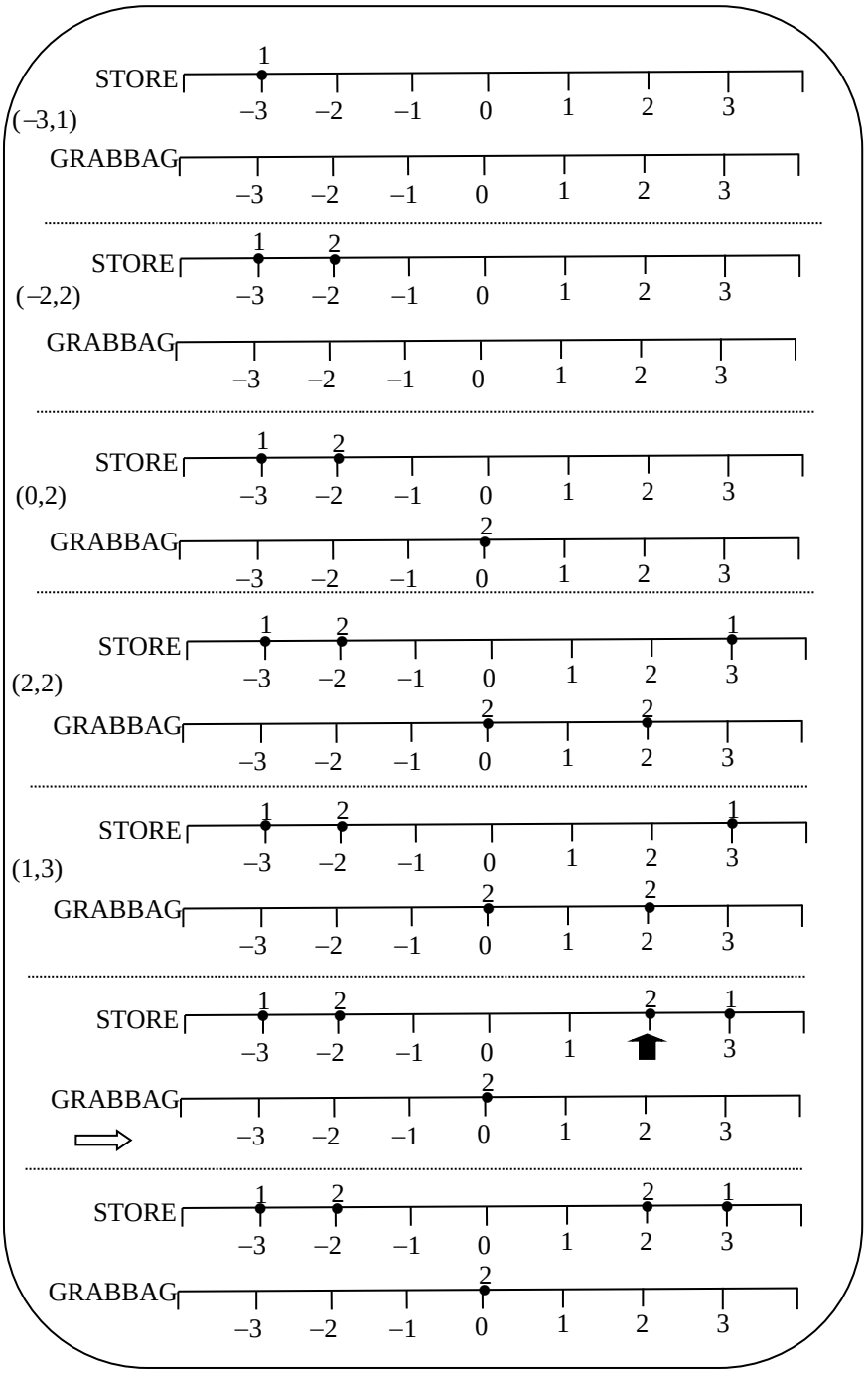


Figura 5.6 Obtenção do conjunto reduzido minimal pelo CNN.

A Figura 5.6 mostra um *trace* do CNN no mesmo conjunto de instâncias usado na Seção 5.4.2 *i.e.*, T_{NN} = {(0,2), (−3,1), (−2,2), (3,1), (2,2)} em que as cinco instâncias são apresentadas ao algoritmo na ordem em que comparecem, em uma reescrita do conjunto de entrada como T_{NN} = {(−3,1), (−2,2), (0,2), (2,2), (3,1)}. Ao final da execução do algoritmo CNN, o conjunto construído em STORE é o T_{CNN} que, nesse exemplo, é o conjunto $T_{minimal}$ = {(−3,1), (−2,2), (2,2), (3,1)}.

5.6 Algoritmo Method$_2$

Na referência [Tomek 1976] são propostos dois algoritmos, o Method1 e o Method2, que, segundo o autor, produzem melhores resultados que o CNN. Esta seção comenta sobre o Method$_1$ e discute o Method$_2$.

5.6.1 Considerações Iniciais

Como Tomek comenta em [Tomek 1976], o algoritmo CNN escolhe as instâncias de maneira randômica e, ao fazer isso permite (1) a participação, no conjunto reduzido resultante, de instâncias desnecessárias e (2) a participação ocasional de instâncias que são internas (*i.e.*, pertencem ao interior do conjunto), ao invés de instâncias que delimitam fronteiras entre as diferentes classes de instâncias no conjunto. De acordo com o autor, as duas modificações propostas por ele eliminam essas desvantagens, por meio da escolha (para a formação do conjunto reduzido) apenas de instâncias que estão próximas da fronteira da classe a que tais instâncias representam.

Tomek alerta que a desvantagem do CNN é causada pelo fato do algoritmo processar as instâncias de T_{NN} de maneira randômica, quando da construção do conjunto reduzido de instâncias, T_{CNN}. Esse fato permite que o T_{CNN}:

(1) contenha instâncias que pertencem ao interior de uma determinada região, que facilmente poderiam ser eliminadas, sem provocar alteração da performance (quando da classificação usando o T_{NN});

(2) contenha instâncias que definem as fronteiras em T_{CNN} mas não em T_{NN} (*i.e.*, instâncias que não são essenciais em T_{NN}, se tornam instâncias de fronteira em T_{CNN}).

A situação (1) implica que o conjunto T_{CNN} é, usualmente, maior do que o necessário e a (2) causa uma mudança indesejável entre fronteiras. Como sugere Tomek, um algoritmo ideal de redução do conjunto T_{NN} deveria implementar o algoritmo CNN,

mas considerar apenas instâncias perto das fronteiras de decisão, ao criar o conjunto reduzido T_{CNN}. Acrescenta, entretanto, que fronteiras de decisão em conjunto de instâncias são desconhecidas.

Uma próxima tentativa seria a de usar apenas aquelas instâncias que geram fronteiras lineares de decisão (*piecewise linear decision boundary*), em T_{NN} e, ainda assim, essa solução é difícil de ser implementada. Tomek comenta que, embora os dois algoritmos propostos por ele sejam bem aquém de serem ideais, podem ser considerados melhoramentos, quando comparados com o CNN. Ambos são baseados em aproximações da noção intuitiva de instância de fronteira.

Apenas o algoritmo $Method_2$ tem recebido atenção em trabalhos na área. Durante o levantamento bibliográfico realizado durante a pesquisa descrita nesta dissertação, apenas uma única descrição do algoritmo $Method_1$ foi encontrada (aquela apresentada em [Tomek 1976]) que, entretanto, apresenta erros tipográficos e, mais importante, a indefinição de uma variável (F) usada em sua descrição; tanto o algoritmo quanto o texto que o comenta não informam como tal variável deve ser inicializada. Esse fato compromete o entendimento e, consequentemente, uma possível implementação do algoritmo $Method_1$.

É importante lembrar que apesar do algoritmo $Method_2$ ter sido o escolhido, ele também tem problemas, que foram identificados após um levantamento bibliográfico mais refinado e que serão discutidos oportunamente nesta dissertação. Os dois algoritmos entretanto, apenas diferem do CNN com relação ao conjunto de instâncias que consideram como input ao algoritmo; apenas instâncias que satisfazem certas propriedades são consideradas por ambos, para a construção do conjunto reduzido.

O $Method_2$ tem por objetivo apenas pré-processar o conjunto inicial de instâncias para, em seguida, fornecê-lo como input ao CNN (reapresentado no Algoritmo 5.4, como descrito em [Tomek 1976]) . Para facilitar o entendimento e promover facilidade de comparação, nesta seção é adotada a notação empregada por Tomek, em [Tomek 1976].

A formalização de um problema de classificação não-paramétrica assume que esteja disponível um conjunto de vetores de valores de atributos, extraídos a partir de um conjunto de objetos. Tal conjunto é geralmente representado por $\{X, \Theta\} = \{(X_1,\theta_1), (X_2,\theta_2), (X_3,\theta_3), ..., (X_N,\theta_N)\}$, em que X_i e θ_i denotam, respectivamente, o vetor de valores de atributos extraídos do i-ésimo objeto e o valor da classe do i-ésimo objeto. Assume-se que os valores de classe estejam corretos e que são extraídos do conjunto de

inteiros {1, 2, ..., M} *i.e.*, as instâncias de X pertencem a uma das M possíveis classes existentes.

Seja D o conjunto original de instâncias, {X, Θ} e seja E o conjunto original condensado $\{X, \Theta\}_E$. O que segue agora é uma reescrita, na notação de Tomek, do funcionamento do CNN. A ideia básica é escolher uma instância arbitrária de D e colocá-la em um conjunto E que tenha sido inicializado anteriormente como um conjunto vazio. As instâncias restantes em D são classificadas pela regra NN (*nearest-neighbour*) usando E e, aquelas que são classificadas incorretamente, são adicionadas a E. Esse procedimento é iterativo até que não aconteçam mais transferências de D para E.

```
procedure cnn            %descrita por Tomek [Tomek 1976]

begin
(a) pass ← 1,
(b) choose x ∈ D randomly, D(1) = D − {x}, E = {x},
(c) D(pass + 1) = ∅, count ← 0,
(d) choose x ∈ D(pass) randomly, classify x by the NN-rule using E
(e) if classification in (d) is correct,
      then D(pass+1) = D(pass + 1) ∪ {x}
      else E = E ∪ {x}, count ←  count + 1,
(f) D(pass) = D(pass) − {x}
(g) if D(pass) ≠ ∅ go to (d)
(h) if count = 0
       then exit
      else pass ← pass + 1, go to (c)
return T_CNN   % E
end procedure
```

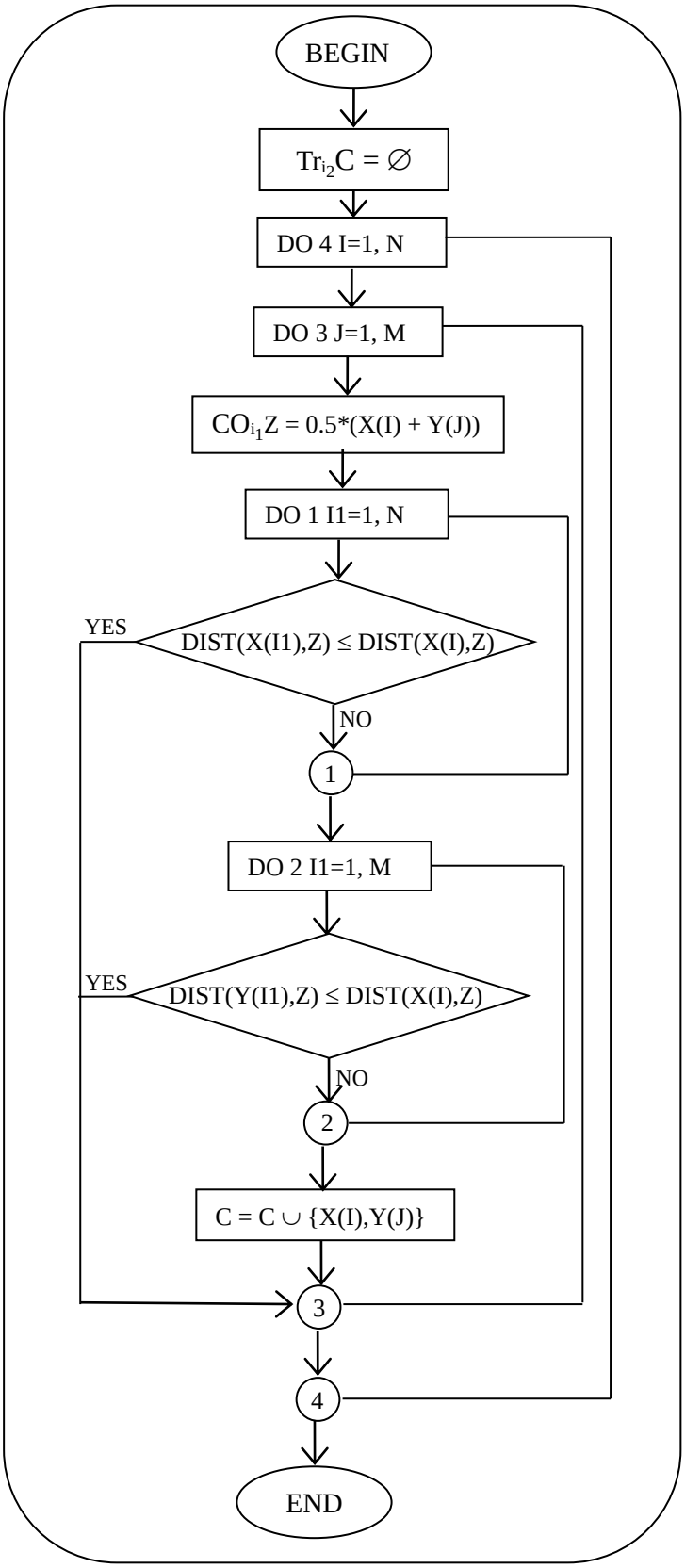
Algoritmo 5.4 Reescrita do algoritmo CNN (Condensed *Nearest Neighbour*) como proposto em [Tomek 1976], já com a correção do erro apontado em [Toussaint 1994]. Na parte else do if em (h), na versão apresentada em [Tomek 1976] está **go to** (b) quando deveria estar **go to** (c), para o algoritmo corresponder ao CNN como proposto em [Hart 1968].

5.6.2 O Algoritmo Method$_2$

Como comentado anteriormente, a modificação do CNN identificada como Mehotd$_2$, proposta por Tomek [Tomek 1976], consiste apenas no pré-processamento do conjunto de instâncias original, referenciado por Tomek como D. O conjunto pré-processado é então fornecido como entrada ao CNN original, para a obtenção do conjunto reduzido.

No pré-processamento do conjunto D para a obtenção de um conjunto reduzido de D, nomeado de C (C ⊂ D), assume-se que X(i), i=1, ..., N são instâncias de D que

pertencem à classe 1 e que Y(i), i = 1, ..., M são instâncias de D que pertencem à classe 2. O fluxograma extraído de [Tomek 1976] e referenciado nessa dissertação como Algoritmo 5.5 segue a sintaxe da linguagem de programação FORTRAN IV e está particularizado para conjuntos de instâncias com duas classes apenas: as N instâncias de classe 1, notadas por X(i), i=1, ..., N e as M instâncias da classe 2, notadas por Y(i), i=1, ..., M. Obviamente |D| = N + M.



Algoritmo 5.5 Pre-procesamento do conjunto original de instâncias ($Method_2$) proposto por Tomek (fluxograma adaptado daquele apresentado em [Tomek 1976]).

O algoritmo Method$_2$ busca no conjunto de treinamento original aquelas instâncias de dados que definem *links* de Tomek. Um *link* de Tomek une instâncias que pertencem a classes diferentes e, de certa forma, definem instâncias que estão na fronteira da classe à qual pertencem.

Considerando um conjunto de instâncias que definem duas classes, classe 1 e classe 2, cada uma das duas instâncias X e Y, que participa de um link de Tomek (X,Y) deve ser tal que X é a vizinha mais próxima de Y e Y deve ser a vizinha mais próxima de X, sendo ambas com classes distintas, uma da outra. O link de Tomek é definido, portanto, por duas instâncias de fronteira, com classes distintas, que são as vizinhas mais próximas uma da outra. A Figura 5.7 exemplifica o conceito de *link de Tomek*, definido por um par de instâncias (X,Y), de classes distintas, em que X pertence à classe • e Y pertence à classe ∗.

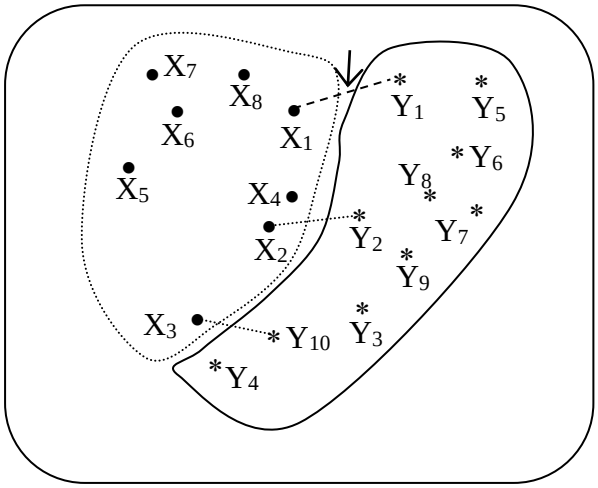


Figura 5.7 Conjunto de treinamento original contendo 18 instâncias, 8 de classe • e 10 de classe ∗. Note que entre os três *links* assinalados, apenas o *link* tracejado e apontado com a flecha é um *link* de Tomek.

Considere o conjunto de treinamento que representa instâncias de duas classes, abordado como dois subconjuntos: os com classe •, {X_1, ..., X_8}, e os com classe ∗, {Y_1, ..., Y_{10}}.

O *link* tracejado na Figura 5.7 (indicado com uma flecha) é um link de Tomek uma vez que X_1 é o vizinho (com classe •) mais próximo de Y_1 (que é de classe ∗) e Y_1 (com classe ∗) é o vizinho mais próximo de X_1 (que é de classe •). Já o link pontilhado definido por (X_2, Y_2) não é link de Tomek pois embora Y_2 seja o vizinho (com classe ∗) mais próximo de X_2, X_2 não é o vizinho com classe • mais próximo de Y_2 – o vizinho com classe •, mais próximo de Y_2, no caso, é X_4.

Situação semelhante acontece com o outro link pontilhado, definido pelas

instâncias (X_3, Y_{10}). X_3 é o vizinho com classe • mais próximo de Y_{10} e, no entanto, Y_{10} não é o vizinho com classe ∗ mais próximo de X_3; tal vizinho é o Y_4. Os *links* definidos (X_4,Y_2) e (X_3,Y_4) (não mostrados na Figura 5.7), são dois outros links de Tomek para o conjunto em questão.

No Apêndice do artigo [Tomek 1976] em que ambos, $Method_1$ e $Method_2$ são apresentados e discutidos, Tomek apresenta um teorema (e sua respectiva prova, via indução), cujo enunciado estabelece que: "Todas as instâncias em D são classificadas corretamente pele regra NN usando o subconjunto C gerado pelo $Method_2$".

Toussaint, em [Toussaint 1994], apresenta um contra-exemplo evidenciado que o teorema estabelecido por Tomek não é válido. O contra-exemplo é descrito a seguir, usando a mesma notação empregada por Toussaint.

Considere o conjunto D contendo seis instâncias de dados, como mostradas na Figura 5.8. Seja {p_1, p_2, p_3} as instâncias de classe 1 e {q_1, q_2, q_3} as instâncias de classe 2. As instâncias são mostradas com suas correspondentes coordenadas (x,y). Considere então o Algoritmo 5.6 sendo executado, usando o conjunto D para produzir o conjunto E, que será, então, fornecido ao CNN. A Tabela 5.2 mostra as coordenadas das seis instâncias de dados de D.

Tabela 5.2 Conjunto D com 6 instâncias de dados bidimensionais e respectivas classes, sendo 3 instâncias de classe 1 e 3 de classe 2.

#ID	X	Y	Classe
p_1	–20	–1	1
p_2	–10	2	1
p_3	0	1	1
q_1	20	1	2
q_2	10	–2	2
q_3	0	–1	2

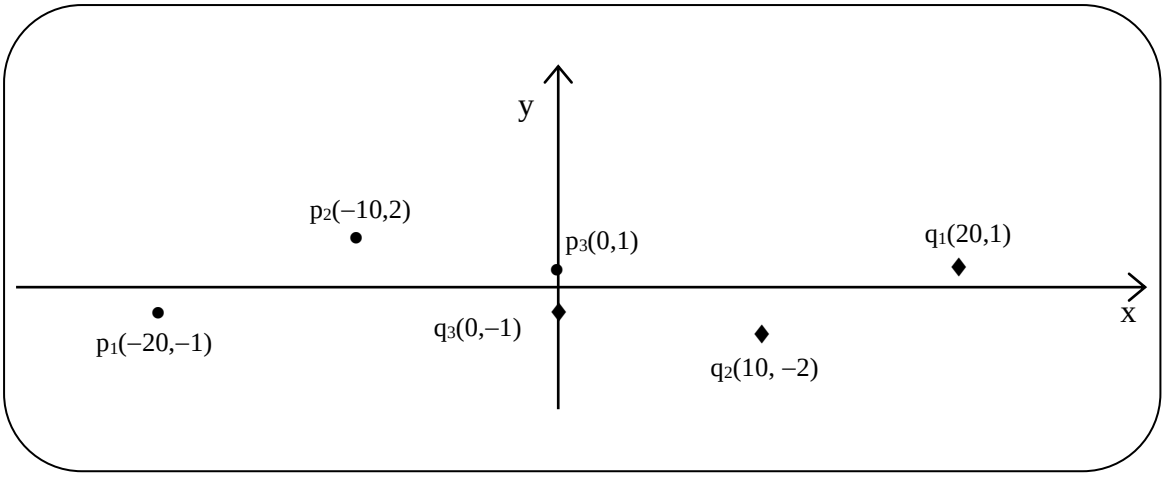


Figura 5.8 Conjunto D com 6 instâncias de dados bidimensionais e respectivas classes, sendo 3 instâncias de classe 1(•) e 3 de classe 2 (♦).

O *trace* do pré-processamento pelo $Method_2$, das 6 instâncias, mostrado na Tabela 5.3, tem como resultado o conjunto reduzido $\{p_3,q_3\}$ i.e., $C = \{p_3,q_3\}$. Fica claro, apenas examinando a Figura 5.8 que, com base em C e usando a regra NN, que tanto p_1 quanto q_1 são classificados incorretamente pelo conjunto C, evidência que C não é um conjunto consistente.

Tabela 5.3 *Trace* do Method$_2$ no conjunto D com 6 instâncias bidimensionais; 3 de classe 1(●) e 3 de classe 2(♦) (Tabela 5.2).

#ID	X	Y	#ID	X	Y	Z	
p_1(●)	–20	–1	q_1(♦)	20	1	0,5*(–20+20) = 0; 0,5*(–1+1) = 0 dis(p_1,Z) = dist(p_1,Z) dis(p_2,Z) ≤ dist(p_1,Z)	(0 0)-
p_1(●)	–20	–1	q_2(♦)	10	–2	0,5*(–20+10)= –5; 0,5*(–1–2) = 0 dis(p_1,Z) = dist(p_1,Z) dis(p_2,Z) ≤ dist(p_1,Z)	(–5 –1,5)
			q_3(♦)	0	–1	0,5*(–20+0)= –10; 0,5*(–1–1) = –1 dis(p_1,Z) = dist(p_1,Z) dis(p_2,Z) ≤ dist(p_1,Z)	(–10 –1)
p_2(●)	–10	2	q_1(♦)	20	1	0,5*(–10+20) = 5; 0,5*(2+1) = 1,5 dis(p_2,Z) = dist(p_2,Z) dis(p_1,Z) ≤ dist(p_2,Z) dis(p_3,Z) ≤ dist(p_2,Z)	(5 1,5)-
p_2(●)	–10	2	q_2(♦)	10	–2	0,5*(–10+10) = 0; 0,5*(2–2) = 0 dis(p_2,Z) = dist(p_2,Z) dis(p_1,Z) ≤ dist(p_2,Z) dis(p_3,Z) ≤ dist(p_2,Z)	(0 0)-
p_2(●)	–10	2	q_3(♦)	0	–1	0,5*(–10+0) = –5; 0,5*(2–1) = 0,5 dis(p_2,Z) = dist(p_2,Z) dis(p_1,Z) ≤ dist(p_2,Z) dis(p_3,Z) ≤ dist(p_2,Z)	(–5 0,5)-
p_3(●)	0	1	q_1(♦)	20	1	0,5*(0+20) = 10; 0,5*(1+1) = 1 dis(p_3,Z) = dist(p_3,Z) dis(p_1,Z) ≤ dist(p_3,Z) dis(p_2,Z) ≤ dist(p_3,Z) dis(q_1,Z) = dist(p_3,Z) dis(q_2,Z) ≤ dist(p_3,Z)	(10 1)
p_3(●)	0	1	q_2(♦)	10	–2	0,5*(0+10) = 5; 0,5*(1–2) = –0,5 dis(p_3,Z) = dist(p_3,Z) (S) dis(p_1,Z) ≤ dist(p_3,Z) dis(p_2,Z) ≤ dist(p_3,Z) dis(q_1,Z) ≤ dist(p_3,Z) (N) dis(q_2,Z) ≤ dist(p_3,Z) (N) dis(q_3,Z) ≤ dist(p_3,Z)	(5 –0,5)
p_3(●)	0	1	q_3(♦)	0	–1	0,5*(0+0) = 0; 0,5*(1–1) = 0 dis(p_3,Z) = dist(p_3,Z) (S) dis(p_1,Z) ≤ dist(p_3,Z) (N) dis(p_2,Z) ≤ dist(p_3,Z) (N) dis(q_1,Z) ≤ dist(p_3,Z) (N) dis(q_2,Z) ≤ dist(p_3,Z) (N) dis(q_3,Z) ≤ dist(p_3,Z)	(0 0)

Capítulo 6
SABI – Sistema para Aprendizado Baseado em Instâncias

6.1 Considerações Iniciais

O *Sistema para Aprendizado Baseado em Instâncias* (SABI) é um sistema computacional desenvolvido com objetivo de disponibilizar um ambiente para a pesquisa relacionada a algoritmos de aprendizado baseado em instâncias, com foco naqueles que se propõem a reduzir o conjunto de dados armazenados.

O SABI foi desenvolvido para plataforma Windows e implementado na linguagem Pascal Object, por meio da ferramenta de desenvolvimento Embarcadero Delphi Tokyo 10.2 (Em: <http://www.embarcadero.com/products/delphi>, acesso em 09 Janeiro 2017) A escolha da ferramenta foi motivada pelo amplo conjunto de componentes visuais que oferece, que facilitam a interação do usuário com o sistema, possibilitando a criação de uma interface intuitiva e de fácil manuseio.

Para que o sistema obtenha um melhor desempenho, o manuseio do conjunto de dados utiliza o banco de dados SQLite (Em: <https://www.sqlite.org>, acesso em 09 Janeiro 2017), por ser um software que não necessita de servidor e ser autossuficiente, no sentido de não fazer uso de bibliotecas externas ou de bibliotecas especificas de qualquer sistema operacional sob o qual está sendo executado.

Com a utilização do banco de dados SQLite, o SABI pode ser utilizado em qualquer computador que possua o sistema operacional Windows XP ou posterior, por não necessitar instalação de qualquer arquivo ou biblioteca, podendo ser utilizado até por meio de dispositivos de memória externa (Pen-Drive, HD-Externo, entre outros).

6.2 Organização e Funcionalidades do SABI

O sistema SABI disponibiliza a implementação de seis algoritmos de aprendizado baseado em instâncias, a saber: NN, IB1, IB2, CNN, RNN e $Method_2$.

Os dados fornecidos para o treinamento (ou teste) do algoritmo a ser utilizado seguem a estrutura CSV (*Comma-Separated Values*), em que cada instância de treinamento deve ser descrita por uma única linha, com os valores que a descrevem

separados por vírgulas, sendo a classe da instância o último valor fornecido. Na primeira linha devem ser fornecidos os nomes dos atributos que descrevem as instâncias, bem como o nome da classe. Os valores ausentes de atributo são representados pelo caractere “?”. Um exemplo da estrutura CSV é apresentado na Figura 6.1.

```
age,sex,cp,trestbps,chol,fbs,restecg,thalach,exang,oldpeak,slope,ca,thal,Class
28,1,2,130,132,0,2,185,0,0,?,?,?,P
29,1,2,120,243,0,0,160,0,0,?,?,?,P
29,1,2,140,?,0,0,170,0,0,?,?,?,P
30,0,1,170,237,0,1,170,0,0,?,?,6,P
31,0,2,100,219,0,1,150,0,0,?,?,?,P
32,0,2,105,198,0,0,165,0,0,?,?,?,P
32,1,2,110,225,0,0,184,0,0,?,?,?,P
32,1,2,125,254,0,0,155,0,0,?,?,?,P
33,1,3,120,298,0,0,185,0,0,?,?,?,P
34,0,2,130,161,0,0,190,0,0,?,?,?,P
34,1,2,150,214,0,1,168,0,0,?,?,?,P
34,1,2,98,220,0,0,150,0,0,?,?,?,P
```

Figura 6.1 Padrão do arquivo de dados (treinamento ou teste) CSV (*Comma-Separated Values*) esperado por qualquer dos algoritmos disponibilizado sob o SABI. A primeira linha informa sequencialmente o nome dos atributos que descrevem os dados, separados por vírgulas (,). As demais linhas descrevem uma instância de dado (treinamento ou teste). Os valores de atributos representados pelo símbolo interrogação (?) devem ser entendidos como valores ausentes (*i.e.*, sem qualquer valor associado).

O SABI disponibiliza dois módulos de processamento: *Pré-Processador* e *Validação da Expressão do Conceito*, que serão descritos nas Seções 6.2 e 6.3 respectivamente.

6.3 Modulo Pré-Processador

O modulo Pré-Processador permite que o usuário possa optar como as instâncias do conjunto de treinamento serão fornecidas para o algoritmo; as duas opções disponibilizadas são a randômica e a sequencial.

Para iniciar a utilização, o usuário precisa selecionar, sob o SABI, (1) qual algoritmo (dentre os disponibilizados pelo sistema) pretende utilizar, (2) como o conjunto de treinamento será processado (de maneira randômica ou sequencial), bem como (3) informar ao sistema o nome de um arquivo do tipo CSV, com os dados de treinamento (fase de treinamento). Para cada instância o SABI atribui um identificador que é utilizado para manipulação dos conjuntos de treinamento e teste (atributo ID). A

Figura 6.2 exibe a tela do SABI após o arquivo de treinamento ter sido carregado.

ID	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak
1	63	1	1	145	233	1	2	150	0	2.3
2	67	1	4	160	286	0	2	108	1	1.5
3	67	1	4	120	229	0	2	129	1	2.6
4	37	1	3	130	250	0	0	187	0	3.5
5	41	0	2	130	204	0	2	172	0	1.4

Figura 6.2 Informações relacionadas ao Módulo Pré-Processador. Na figura, o item (1) em vermelho indica o campo em que é feita a seleção do algoritmo a ser utilizado; o item (2) indica o local em que a seleção da forma de processamento das instâncias do conjunto de dados deve ser realizada; o item (3) indica o campo em que a seleção do arquivo com o conjunto de instâncias deve ser feita; (4) Abas do sistema SABI onde são exibidas as instâncias em seus respectivos processo; o item (5) mostra os valores mínimos e máximos de cada atributo e a quantidade de atributos ausentes, com relação aos dados carregados e, finalmente, o item (6) indica a área de exibição das instâncias.

Considerando que o arquivo de treinamento esteja carregado em memória, a avaliação do conceito (que no caso é representado pelas instâncias em memória) é feita na fase de classificação, usando instâncias de teste, que podem ser fornecidas de duas maneiras como descritas em (A) e (B) a seguir.

(A) *Entrada em lote* – o usuário seleciona, sob o SABI, um arquivo do tipo CSV, com as instâncias de teste. Este arquivo contém instâncias previamente classificadas (via de regra tais instâncias representam um fragmento do conjunto original de dados que, presume-se, estejam corretamente classificadas). O sistema vai então contabilizar, comparando a classe fornecida associada a cada instância, com aquela inferida pela expressão do conceito armazenada em memória, quantos acertos/erros de classificação o conceito em questão fez. A Figura 6.3 mostra a tela apresentada pelo SABI, com parte do processo.

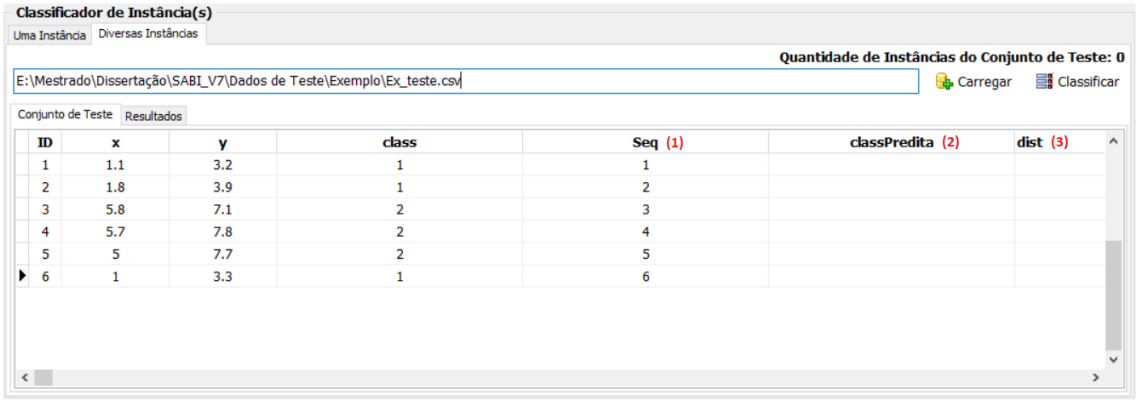


Figura 6.3 Classificador de instâncias – Entrada em lote arquivo CSV. Na figura, o item (1), apresenta a sequência que as instâncias serão classificadas; o item (2), classPredita irá receber a classe determinada pelo sistema SABI; o item (3), a distância da instância que fez a classificação.

(B) *Entrada manual de instâncias sem classe associada* – o usuário pode desejar que uma (informada diretamente na tela) ou várias instâncias (informada via arquivo CSV) sejam classificadas. Nessa opção o sistema exibe, de forma dinâmica, os atributos que devem ter seus respectivos valores digitados, tendo por base a quantidade de atributos do arquivo de treinamento. Na Figura 6.4 é mostrada a entrada manual de uma instância.

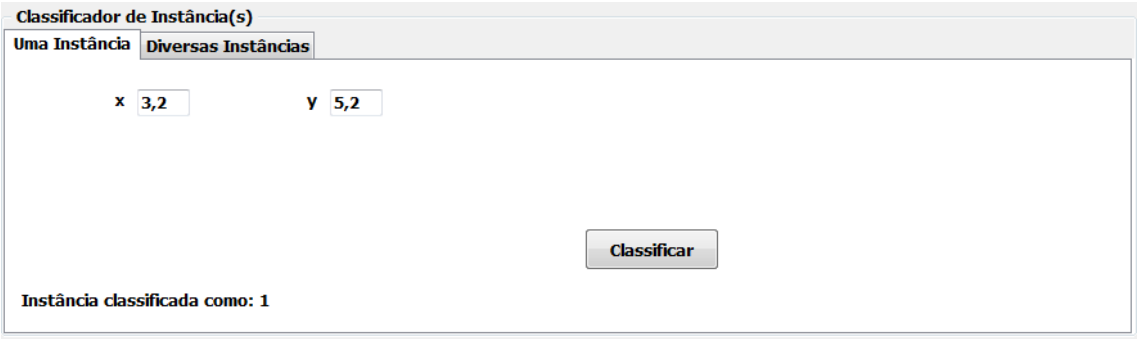


Figura 6.4 Classificador de instâncias – Área de digitação de uma instância a ser classificada.

O sistema apresenta as instâncias classificadas, com suas respectivas classes associadas. Para facilitar a visualização dos resultados quando as instâncias já possuírem uma classificação, o SABI colore a instância com verde quando a classificação for correta e com vermelho, quando for incorreta, como mostra a Figura 6.5.

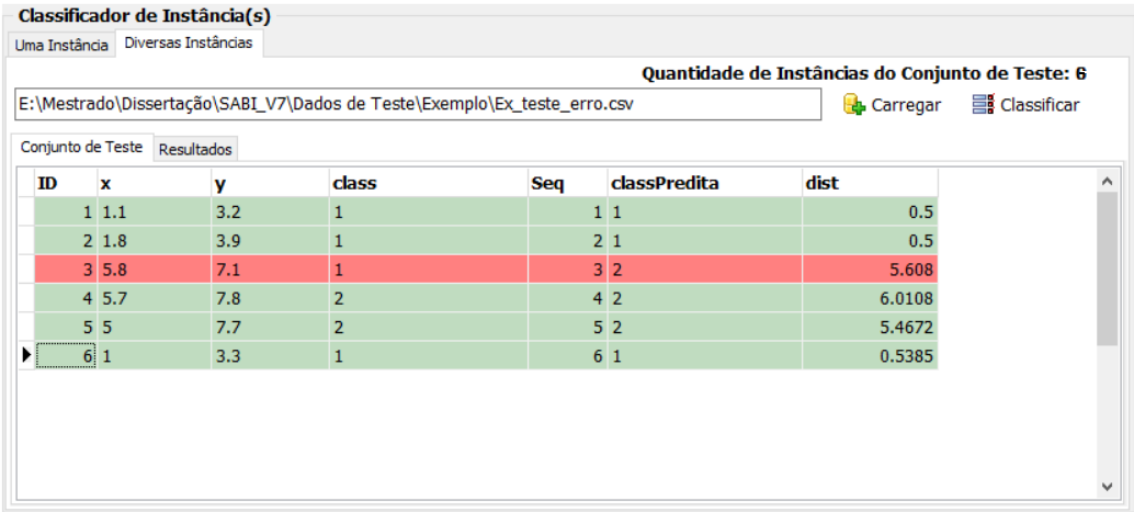


Figura 6.5 Saída de dados – Apresentação das instâncias, com sua respectiva classificação. As cores verde e vermelho indicam, respectivamente, classificação correta e incorreta.

O sistema apresenta um resumo do processo de classificação, na aba Resultados, como mostra a Figura 6.6. Nela são mostrados o algoritmo que foi utilizado para a classificação, o número de instâncias de treinamento, o número de instâncias de teste e o número de acertos/erros de classificação.

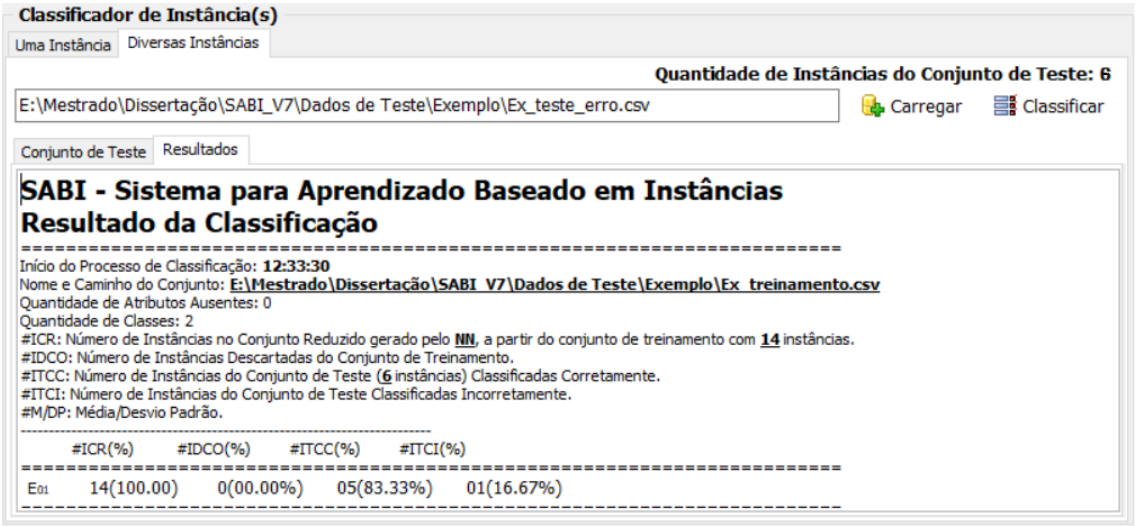


Figura 6.6 Saída de dados – Resumo do resultado da classificação.

Para uma melhor visualização, o sistema disponibiliza uma interface gráfica simples, em que dois atributos que descrevem as instâncias devem ser selecionados pelo usuário (via tela), e uma plotagem das instâncias (descritas pelos atributos selecionados) é exibida, com as respectivas classes. O usuário pode escolher exibir ou não os valores de cada instância, podendo também selecionar quais instâncias serão exibidas no gráfico (Treinamento, Teste, Treinamento/Teste), como mostrado na Figura 6.7.

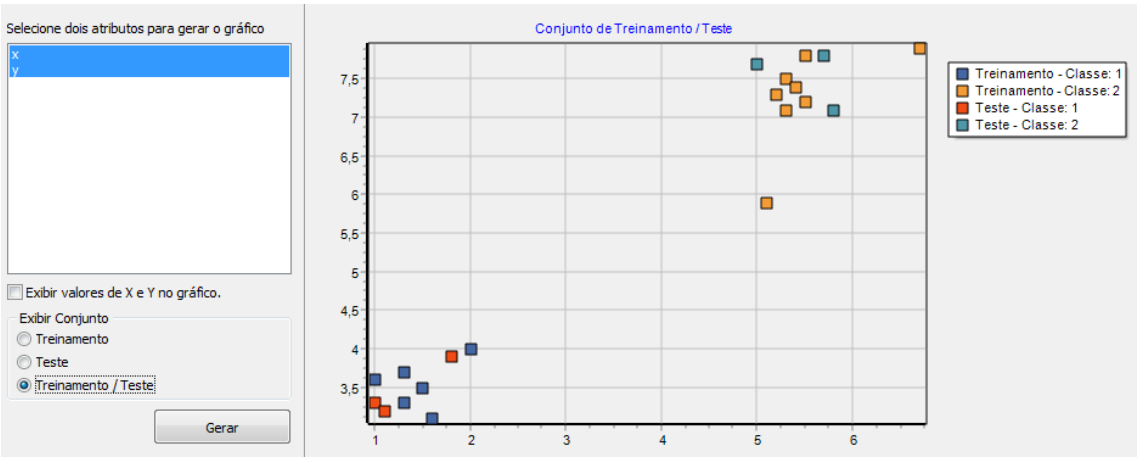


Figura 6.7 Saída de dados – Via plotagem dos dados.

6.4 Validação da Expressão do Conceito

O modulo de *Validação da Expressão do Conceito* permite a criação de um ambiente para a realização de experimentos, sob o qual o usuário pode informar o número de pares de conjuntos Treinamento-Teste pretende utilizar. Os pares são automaticamente criados pelo SABI, selecionando randomicamente as instâncias que comporão os conjuntos Treinamento-Teste, a partir do conjunto original fornecido.

O número de instâncias em cada um dos conjuntos de cada par é determinado pelo usuário. Por *default*, o sistema SABI está configurado para que 80% das instâncias pertençam ao conjunto de treinamento e os 20% restantes ao conjunto de teste (a porcentagem *default*, entretanto, pode ser alterada pelo usuário). A Figura 6.8 apresenta uma enumeração dos passos que o usuário deve seguir para a realização do experimento.

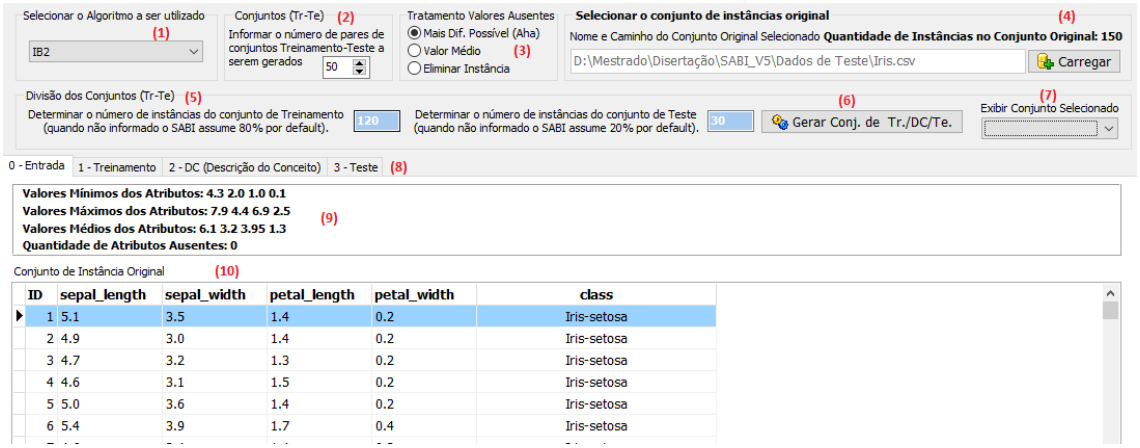


Figura 6.8 Criação do ambiente para experimento. Ação: (1) Selecionar o algoritmo a ser utilizado. (2) Informar a quantidade de pares de Treinamento/Teste a serem gerados. (3) Selecionar o tratamento de valores ausentes. (4) Selecionar arquivo com o conjunto de instâncias. (5) Quantidades de instâncias que o conjunto de Treinamento e Teste irão possuir. (6) Pressionar o botão que inicia a fase de treinamento. (7) Selecionar o par de conjuntos a serem exibidos. (8) Abas do SABI que exibem os conjuntos de instâncias. (9) Painel com os valores mínimos, médio e máximos de cada atributo e com o número de atributos com valores ausentes. (10) Área de exibição das instâncias.

Ao termino da execução da fase de treinamento o sistema SABI aguarda que o usuário clique sobre o botão *Classificar* para iniciar a fase de classificação. Como comentado anteriormente, quando da classificação correta de uma instância, o SABI exibe a instância em área com fundo verde e, quando da classificação incorreta, em área com fundo vermelho. A Figura 6.9 apresenta a tela de Teste.

Figura 6.9 Classificação das instâncias. (1) Aba para exibição das classificações das instâncias de teste. (2) Aba contendo as instâncias do Conjunto de Teste. (3) Aba em que são apresentados os resultados da classificação. (4) Botão Classificar. (5) Botão para Exportação dos Resultados em um arquivo no formato RTF.

Como resultado final o SABI apresenta um resumo do domínio de dados utilizado no experimento, apresentado a hora do início do processo de classificação, o nome do arquivo de dados fornecido para o experimento, o número de valores ausentes, o número de classes, identificação do algoritmo utilizado, o número total de instâncias e o

número de instâncias para cada um dos conjuntos do par de conjuntos Treinamento–Teste.

Considerando cada par de conjuntos Treinamento–Teste, é criado um identificador para o Experimento (E_{01},..., E_n), e são exibidos para cada um deles o número de instâncias e os seguintes percentuais: (#IRC) Instâncias no Conjunto Reduzido, (#IDCO) Instâncias no Conjunto Descartado, (#ITCC) Instâncias Classificadas Corretamente e (#ITCI) Instâncias Classificadas Incorretamente.

Ao final do Experimento são exibidas as médias e desvio padrão associado para os valores comentados acima. A Figura 6.10 e a Figura 6.11 exibem a tela de Resultados.

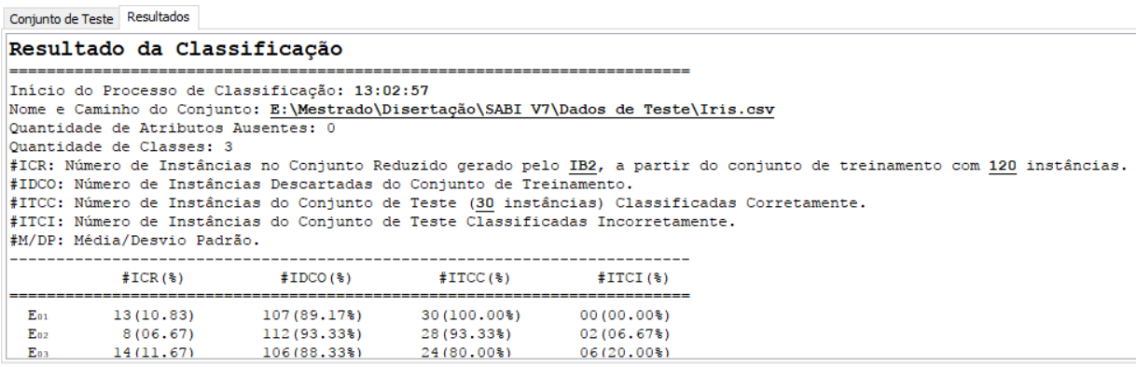


Figura 6.10 Resultado da Classificação – Resumo do domínio de dado utilizado no Experimento.

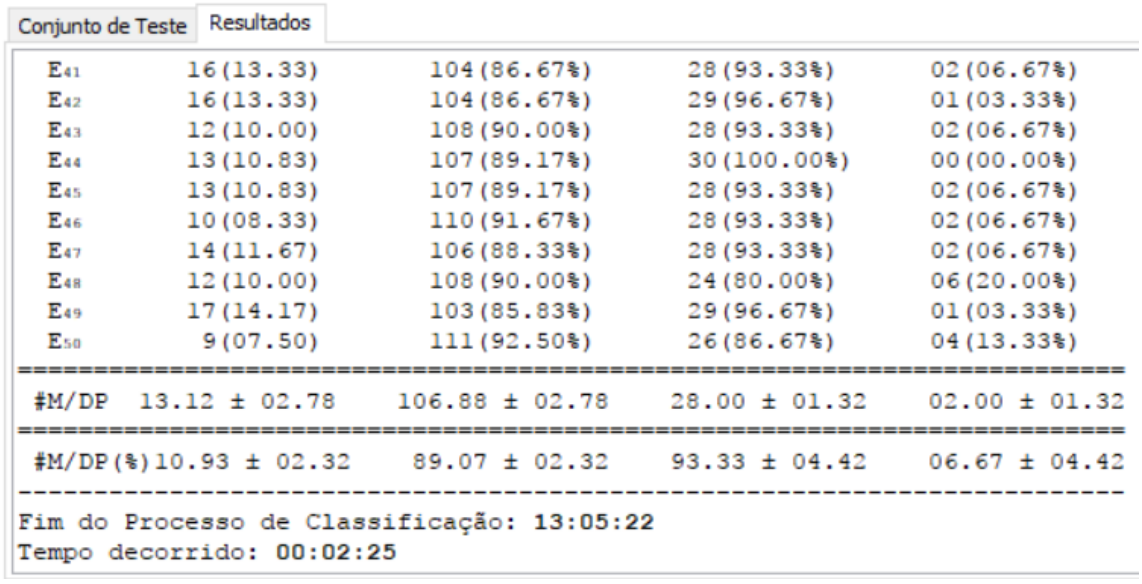
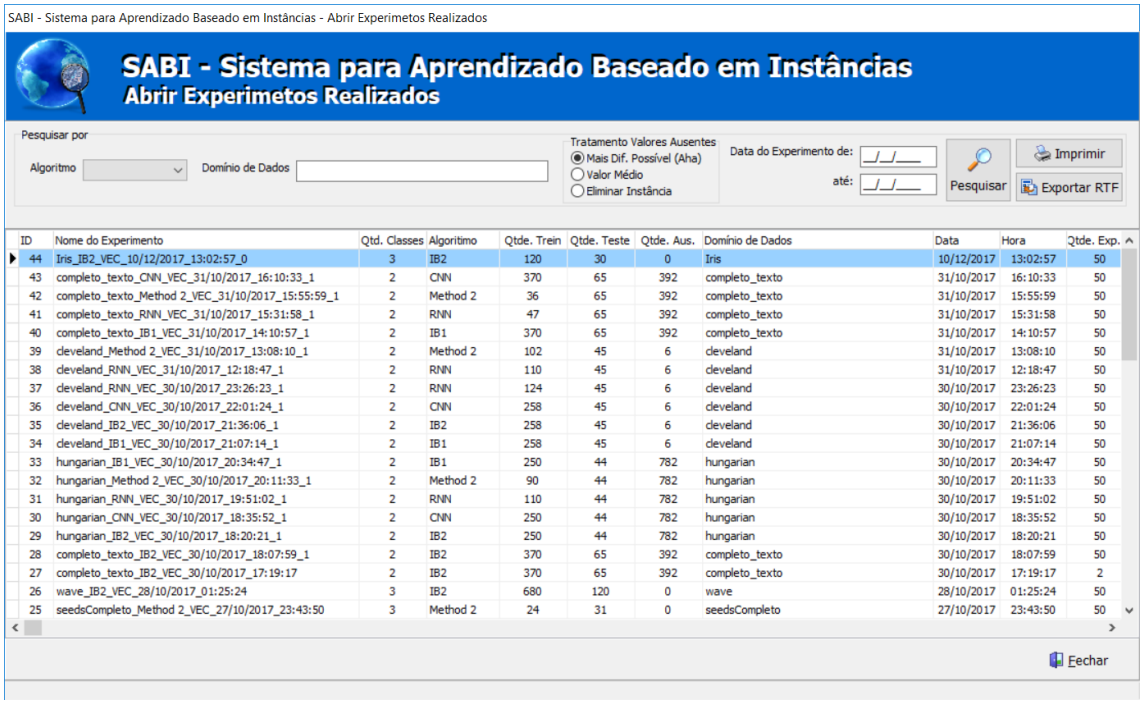


Figura 6.11 Resultado da Classificação – Sumarização dos valores obtidos no Experimento.

6.5 Exibição e Armazenamento dos Resultados

O Sistema SABI armazena informações e resultados relacionados a todos os experimentos realizados, para uma eventual consulta e recuperação de informações associadas a determinados experimentos. O SABI possibilita que os experimentos sejam pesquisados por algoritmo, domínio de dados, tipo de tratamento de valores ausentes e por data em que o experimento foi realizado. A Figura 6.12 apresenta a Tela de Abrir Experimentos Realizados.



ID	Nome do Experimento	Qtd. Classes	Algoritimo	Qtde. Trein	Qtde. Teste	Qtde. Aus.	Domínio de Dados	Data	Hora	Qtde. Exp.
44	Iris_IB2_VEC_10/12/2017_13:02:57_0	3	IB2	120	30	0	Iris	10/12/2017	13:02:57	50
43	completo_texto_CNN_VEC_31/10/2017_16:10:33_1	2	CNN	370	65	392	completo_texto	31/10/2017	16:10:33	50
42	completo_texto_Method 2_VEC_31/10/2017_15:55:59_1	2	Method 2	36	65	392	completo_texto	31/10/2017	15:55:59	50
41	completo_texto_RNN_VEC_31/10/2017_15:31:58_1	2	RNN	47	65	392	completo_texto	31/10/2017	15:31:58	50
40	completo_texto_IB1_VEC_31/10/2017_14:10:57_1	2	IB1	370	65	392	completo_texto	31/10/2017	14:10:57	50
39	cleveland_Method 2_VEC_31/10/2017_13:08:10_1	2	Method 2	102	45	6	cleveland	31/10/2017	13:08:10	50
38	cleveland_RNN_VEC_31/10/2017_12:18:47_1	2	RNN	110	45	6	cleveland	31/10/2017	12:18:47	50
37	cleveland_RNN_VEC_30/10/2017_23:26:23_1	2	RNN	124	45	6	cleveland	30/10/2017	23:26:23	50
36	cleveland_CNN_VEC_30/10/2017_22:01:24_1	2	CNN	258	45	6	cleveland	30/10/2017	22:01:24	50
35	cleveland_IB2_VEC_30/10/2017_21:36:06_1	2	IB2	258	45	6	cleveland	30/10/2017	21:36:06	50
34	cleveland_IB1_VEC_30/10/2017_21:07:14_1	2	IB1	258	45	6	cleveland	30/10/2017	21:07:14	50
33	hungarian_IB1_VEC_30/10/2017_20:34:47_1	2	IB1	250	44	782	hungarian	30/10/2017	20:34:47	50
32	hungarian_Method 2_VEC_30/10/2017_20:11:33_1	2	Method 2	90	44	782	hungarian	30/10/2017	20:11:33	50
31	hungarian_RNN_VEC_30/10/2017_19:51:02_1	2	RNN	110	44	782	hungarian	30/10/2017	19:51:02	50
30	hungarian_CNN_VEC_30/10/2017_18:35:52_1	2	CNN	250	44	782	hungarian	30/10/2017	18:35:52	50
29	hungarian_IB2_VEC_30/10/2017_18:20:21_1	2	IB2	250	44	782	hungarian	30/10/2017	18:20:21	50
28	completo_texto_IB2_VEC_30/10/2017_18:07:59_1	2	IB2	370	65	392	completo_texto	30/10/2017	18:07:59	50
27	completo_texto_IB2_VEC_30/10/2017_17:19:17	2	IB2	370	65	392	completo_texto	30/10/2017	17:19:17	2
26	wave_IB2_VEC_28/10/2017_01:25:24	3	IB2	680	120	0	wave	28/10/2017	01:25:24	50
25	seedsCompleto_Method 2_VEC_27/10/2017_23:43:50	3	Method 2	24	31	0	seedsCompleto	27/10/2017	23:43:50	50

Figura 6.12 Abrir Experimentos Realizados – Possibilita a pesquisa de experimentos por algoritmo utilizado, domínio de dados utilizado, utilização (ou não) de tratamento de valores ausentes e por data que o experimento foi realizado.

Com o objetivo de disponibilizar um ambiente que facilita a organização/recuperação de experimentos, o SABI cria um identificador associado ao experimento, que é uma cadeia de caracteres composta pelo nome do domínio, algoritmo utilizado, modulo do sistema SABI utilizado (*PP - Pré-Processado* ou *VEC - Validação da Expressão do Conceito*), data e hora da realização do experimento e tratamento do valor ausente utilizado (*0 – Mais Dif. Possível, 1 – Valor Médio, 2 – Eliminação da Instância*).

Os resultados dos experimentos podem ser impressos ou exportados para um arquivo, em formato RTF (*Rich Text Format*). As figuras 6.13 e 6.14 apresentam as telas

de impressão e de exportação, respectivamente.

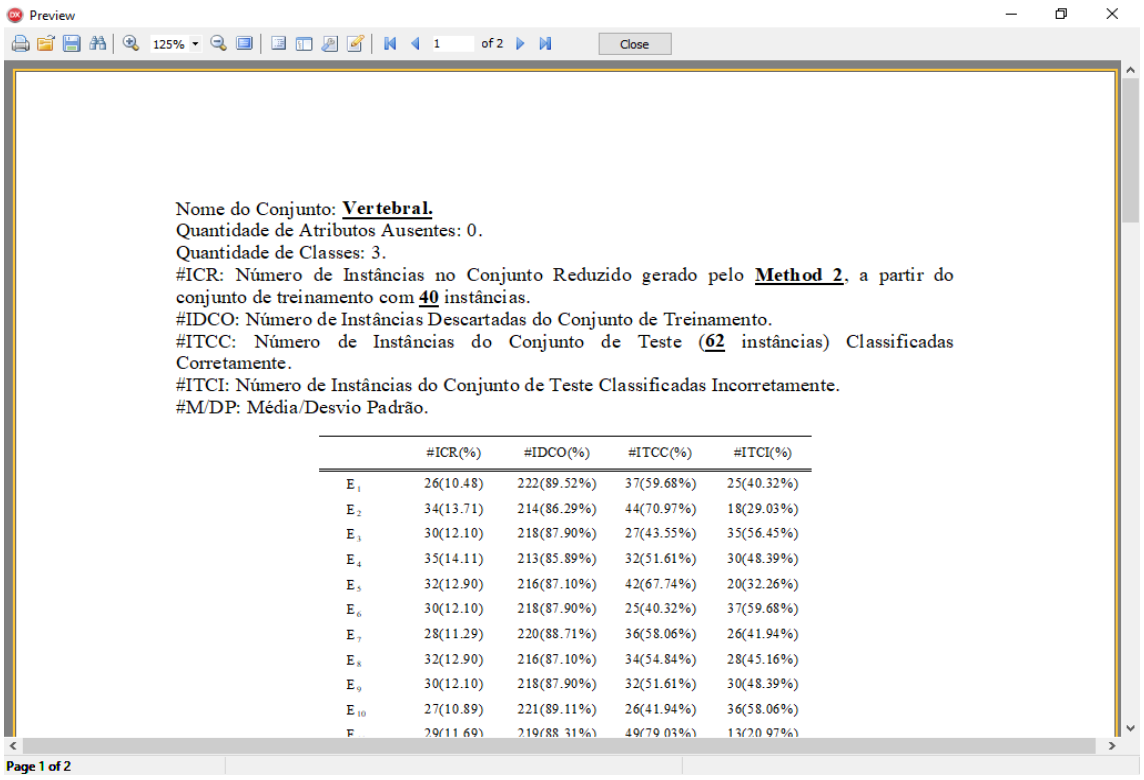


Figura 6.13 Tela de impressão de experimento.

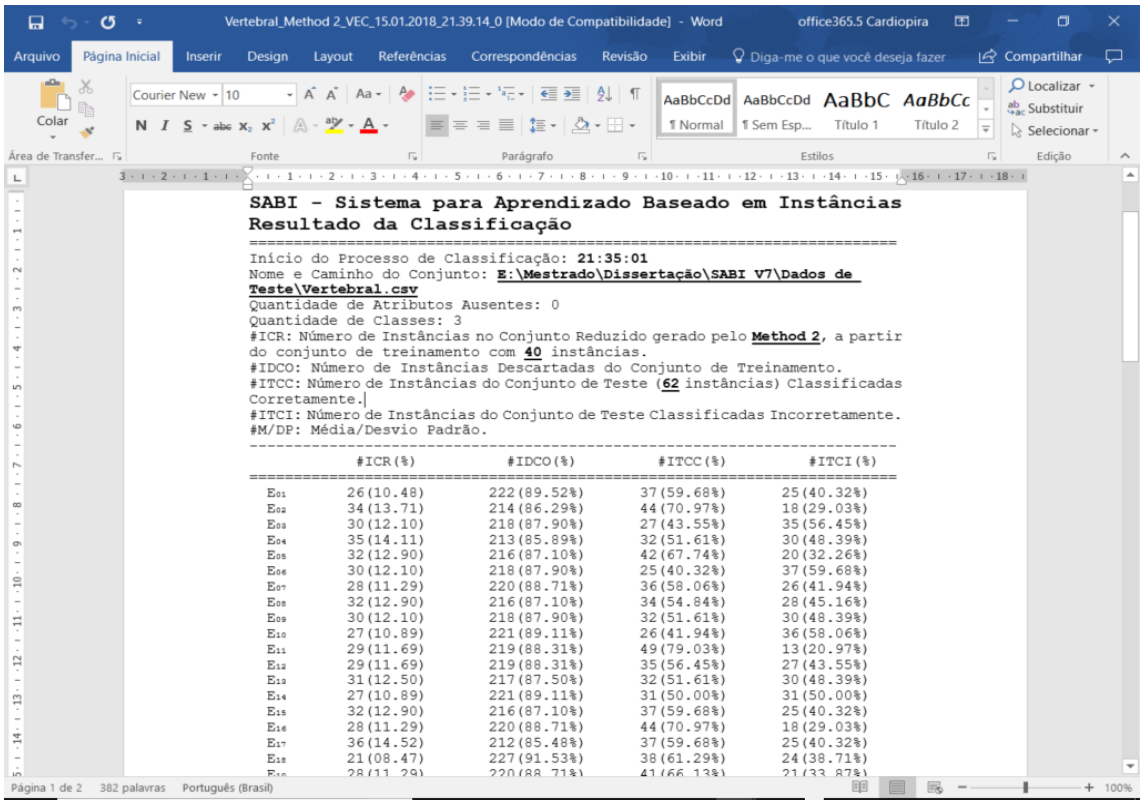


Figura 6.14 Experimento exportado para o formato RTF e aberto em um editor de texto (Microsoft Word).

Capítulo 7
Descrição dos Dados, Metodologia, Experimentos e Discussão dos Resultados

7.1 Considerações Iniciais

Com o objetivo de avaliar a contribuição de estratégias de redução do volume de dados armazenados por algoritmos ABI, um conjunto de experimentos foram conduzidos no ambiente do sistema SABI (*Sistema de Aprendizado Baseado em Instâncias*), projetado e implementado para esse fim, durante o desenvolvimento da pesquisa descrita nesta dissertação. Este capítulo aborda em detalhes os experimentos conduzidos e está organizado em seis seções.

Na Seção 7.2 são apresentados os conjuntos de dados utilizados na parte experimental da pesquisa, bem como descritas suas principais características. A Seção 7.3 tem por foco a descrição do processo de imputação adotado, disponibilizado sob o SABI, para tratar o problema de valores ausentes de atributos, como uma fase anterior ao aprendizado. Na Seção 7.4 é detalhada a metodologia adotada para a realização dos experimentos e a Seção 7.5 apresenta os resultados dos experimentos e uma discussão parcial sobre eles. Com vistas a explorar um pouco mais o uso de imputação de dados, a Seção 7.6 mostra resultados obtidos relativos à redução de volume de dados, em que foram utilizadas duas outras técnicas de imputação de dados. A Seção 7.7 retoma a discussão dos resultados obtidos em todos os experimentos realizados, envolvendo os experimentos das duas seções anteriores.

7.2 Conjuntos de Dados Utilizados nos Experimentos

Para a realização dos experimentos foram escolhidos 10 conjuntos de dados, sendo que 9 deles foram extraídos do UCI Repository [Lichman 2013] e 1 (denominado Mouse), obtido junto a um dos autores do trabalho [Nietto & Nicoletti 2017]. As principais características dos 10 domínios de dados são apresentadas na Tabela 7.1

Tabela 7.1 Características dos domínios de dados utilizados nos experimentos. Na tabela foi usada a notação: #ID: número associado ao domínio; DD: Nome do domínio de dados; #TIO: Número total de instâncias em DD; #AT: número de atributos que descrevem as instâncias; TA: Número de atributos de determinado tipo (C: categórico, I: inteiro, R: real); AA/NVA: existência (S) (ou não (N)) de valores ausentes de atributos/número de valores ausentes; #C: Número de classes existentes; #IC/C/% número de instância na classe/nome da classe/porcentagem de instâncias pertencentes a classe.

#ID	DD	#TIO	#AT	TA	AA/NVA	#C	#IC/C/%
1	Car	1.728	6	6 C	N	4	1,210/unacc/70,023 384/acc/22,222 69/good/3,993 65/v-good/3,762
2	Cleveland	303	13	12 I & 1 R	S/6	2	139/N/45,874 164/P/54,125
3	Hungarian	294	13	12 I & 1 R	S/782	2	106/N/36,054 188/P/63,945
4	Iris	150	4	4 R	N	3	50/setosa/33,33 50/versicolour/33,33 50/virginica/33,33
5	Liver	345	6	5 I & 1 R	N	2	145/1/40,960 209/2/59,039
6	Mouse	1000	2	2R	N	3	200/A/20,00 200/B/20,00 600/C/60,00
7	Seeds	210	7	7 R	N	3	70/1/33,33 70/2/33,33 70/3/33,33
8	Waveform	800	21	21 R	N	3	267/0/33,375 267/1/33,375 266/2/33,25
9	Vertebral	310	6	6 R	N	3	60/Hernia/19,354 150/Spondylolisthesis/48,387 100/Normal/32,258
10	Voting	435	16	16 C	S/392	2	267/democrat/61,379 168/republican/38,620

7.3 Tratamento de Valores Ausentes

Para tentar suavizar o impacto da presença de valores ausentes de atributo em conjuntos de dados, foi empregada a estratégia de imputação de dados proposta em [Aha *et al.* 1991], descrita por meio de um exemplo, como segue.

Considere um espaço 5-dimensional definido pelos atributos A_1, A_2, A_3, A_4 e A_5, cujos valores mínimo e máximo associados a cada um dos cinco atributos, com base no conceito armazenado, sejam como mostra a Tabela 7.2. Considere também que a instância de dado x = (3,2 7,8 4,0 ? 12,9) faça parte da descrição do conceito. Note que a instância x tem um valor ausente (aquele associado ao atributo A_4), indicado pelo presença do símbolo ?.

Tabela 7.2 Valores mínimo e máximo associados a cada um dos cinco atributos da situação hipotética sendo considerada.

Atributo	Valor mínimo	Valor máximo
A_1	1,0	38,9
A_2	5,2	18,6
A_3	3,8	32,3
A_4	4,9	27.6
A_5	11,0	59,2

Suponha que, na fase de classificação, deseja-se saber a que classe a instância y = (5,2 8,6 4,7 6,7 15,9) pertence. Para determinar a classe de y, um algoritmo ABI calcula a distância de y a cada uma das instâncias que participam da descrição do conceito, o que inclui o cálculo da distância de y à instância x.

Como a instância x tem o atributo A_4 com valor ausente, para o cálculo da distância entre x e y, assume-se como valor de A_4 em x o valor mais distante possível do valor de A_4 em y (*i.e.*, o mais distante possível de 6,7, que é o valor de A_4 em y).

Considerando agora os valores mínimo e máximo associados a cada atributo, mostrados na Tabela 7.2, tal valor será 27,6 e, portanto, o algoritmo calcula a distância entre y = (5,2 8,6 4,7 6,7 15,9) e x = (3,2 7,8 4,0 27,6 12,9). Em situações em que os respectivos valores de um determinado atributo estiverem ausentes em ambos, x e y, para o cálculo da distância entre ambos, x e y, assume-se que a distância entre os atributos com valor ausente seja 1.

7.4 Descrição da Metodologia Empregada para a Realização dos Experimentos

A descrição da metodologia empregada para a realização dos experimentos tem duas fases: a primeira é identificada como *pré-processamento dos conjuntos de dados originais*, descrita pelo fluxograma da Figura 7.1, e a segunda diz respeito ao *uso de um dos algoritmos de redução de volume de dados implementado*, nos dados obtidos no pré-processamento (como mostrado em Figura 7.2) e a contabilização do resultado final associado a cada um dos 10 domínios. Tais resultados foram calculados como a média/desvio padrão dos valores de precisão e de taxa de redução de volume nos 50 experimentos associados ao referido domínio, para todos os domínios.

No fluxograma da Figura 7.1, para cada domínio identificado como i, a partir do conjunto original de dados CO_i, são criadas 50 versões desse mesmo conjunto, com o sequenciamento de instâncias feito de maneira randômica. Os conjuntos CO_{i_1}, CO_{i_2},

...$CO_{i_{50}}$, representam, pois, o mesmo conjunto de dados CO_i, mas com as respectivas instâncias de CO_i embaralhadas.

Para cada específico CO_i (i=1, ..., 10), são criados 50 pares de conjuntos, identificados como Tr_{i_j}–Te_{i_j} (j=1, ..., 50), contendo respectivamente 85% e 15% de instâncias de CO_i. A Tabela 7.3 apresenta o número de instâncias de treinamento e o correspondente número de instâncias de teste, de cada um dos 50 pares de conjunto Treinamento(Tr)-Teste(Te) gerados (ver Figura 7.1), associados a cada um dos 10 domínios de dados considerados.

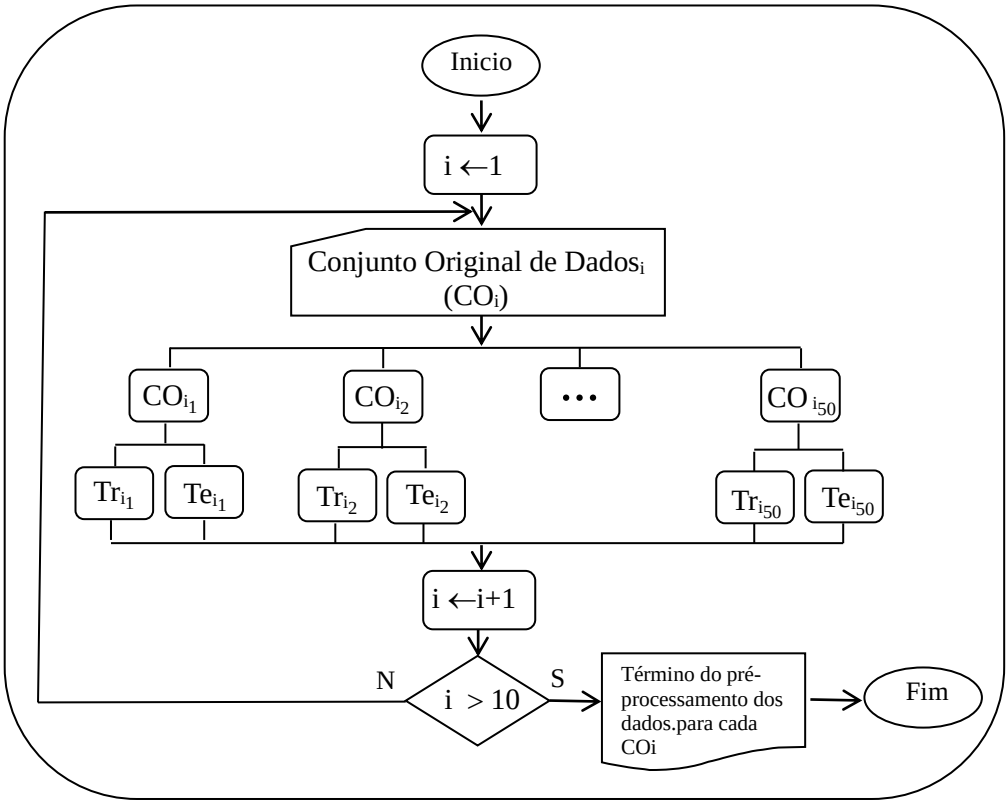


Figura 7.1 Fluxograma do processo de pré-processamento dos 10 arquivos de dados originais, associados a cada um dos domínios de dados, para a obtenção de 50 pares de conjuntos Treinamento (Tr)-Teste(Te) que serão usados como input para cada um dos algoritmos de redução de volume de dados.

Tabela 7.3 DD: domínio de dados; #ID: número de identificação do domínio; #ICO: Número de instâncias no conjunto original; #ITr: Número de instâncias no conjunto de treinamento; #ITe: Número de instâncias no conjunto de teste

DD	#ID	#ICO	#Itr	#ITe
Car	1	1.728	1469 (85,01%)	259 (14,99%)
Cleveland	2	303	258 (85,15%)	45 (14,85%)
Hungarian	3	294	250 (85,03%)	44 (14,97%)
Iris	4	150	128 (85,33%)	22 (14,97%)
Liver	5	345	293 (85,03%)	52 (14,97%)
Mouse	6	1000	850 (85,00%)	150 (15,00%)

Seeds	7	210	179 (85,24%)	31 (14,76%)
Waveform	8	800	680 (85,00%)	120 (15,00%)
Vertebral	9	310	264 (85,16%)	46 (14,84%)
Voting	10	435	370 (85,06%)	65 (14,94%)

No fluxograma da Figura 7.2 cada um dos algoritmos de redução de volume implementado, espera como entrada a identificação do domínio de dados, variável i para, então, acessar os 50 pares de conjuntos de dados Tr-Te, associados ao domínio i em questão, obtidos no processo de pré-processamento de dados, descrito na Figura 7.1. O correspondente algoritmo de redução de volume então cria a representação do conceito usando o conjunto Tr. Na sequência, tal representação é avaliada, usando o conjunto Te correspondente e os resultados de precisão e redução são coletados. Repete o procedimento para todos os 49 pares restantes de Tr-Te e, ao finalizar, contabiliza as médias e desvios-padrão dos valores calculados.

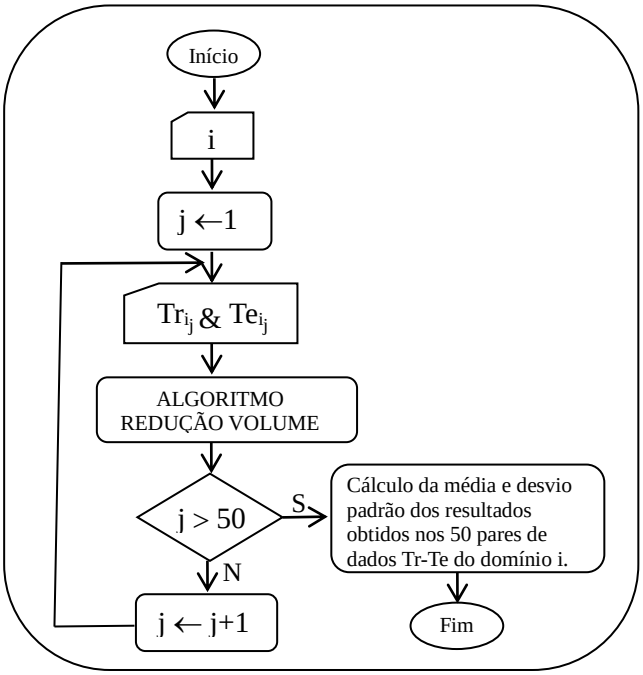


Figura 7.2 Fluxograma do processo de execução de cada um dos quatro algoritmos de redução de volume empregados; cada um deles está sendo referenciado genericamente, no fluxograma, como ALGORITMO REDUÇÃO VOLUME.

7.5 Resultados dos Experimentos Realizados

Esta seção apresenta e discute os resultados obtidos nos experimentos utilizando os quatro algoritmos ABI que foram foco da pesquisa a saber, IB2, CNN, RNN e Method2, bem como os resultados obtidos pelo algoritmo IB1 (algoritmo baseado em instância, que não implementa redução de volume), para efeito de comparação. Como mostrados na Tabela 7.4, por resultados devem ser entendidos: (1) a precisão obtida pelos

algoritmos de redução (número de instâncias do conjunto de teste classificados corretamente) e (2) o percentual de instâncias de dados (do conjunto de dados original) que foi efetivamente armazenado, para cada um dos domínios de dados.

Tabela 7.4 DD: domínio de dados; Colunas IB2, CNN, RNN e $Method_2$: são os algoritmos de redução utilizados nos experimentos e IB1, ABI sem redução de volume, utilizado para comparação; #Cc Precisão de cada algoritmo; #% Porcentagem de Armazenamento.

#DD	IB1		IB2		CNN		RNN		$Method_2$	
	#Pc	#%	#Pc	#%	#Pc	#%	#Pc	#%	#Pc	#%
Car	94,73	100	94,09	13,89	94,47	13,74	93,88	13,76	58,11	06,39
Cleveland	58,76	100	54,58	44,03	55,91	44,57	55,87	32,27	50,53	26,88
Hungarian	60,64	100	57,73	44,54	57,00	45,11	54,32	39,29	55,86	26,78
Iris	95,91	100	93,18	10,31	93,91	06,01	88,91	07,37	51,18	04,61
Liver	62,93	100	57,19	43,45	56,58	43,15	55,96	36,75	54,00	25,98
Mouse	98,68	100	98,20	03,56	97,84	03,45	97,07	02,08	75,44	01,45
Seeds	91,23	100	89,10	15,49	96,90	15,23	85,29	11,39	66,71	09,83
Waveform	73,87	100	69,07	31,86	69,47	32,09	68,43	24,14	64,35	12,90
Vertebral	82,48	100	75,22	26,63	76,35	26,01	74,87	20,16	58,48	11,52
Voting	93,05	100	89,75	11,77	89,66	12,07	89,05	11,25	80,49	06,04

A Figura 7.3 exibe os resultados obtidos, para cada um dos domínios de dados, evidenciando a precisão e o percentual de armazenamento (relativa ao número total de instâncias contidas no conjunto de dados) requerido.

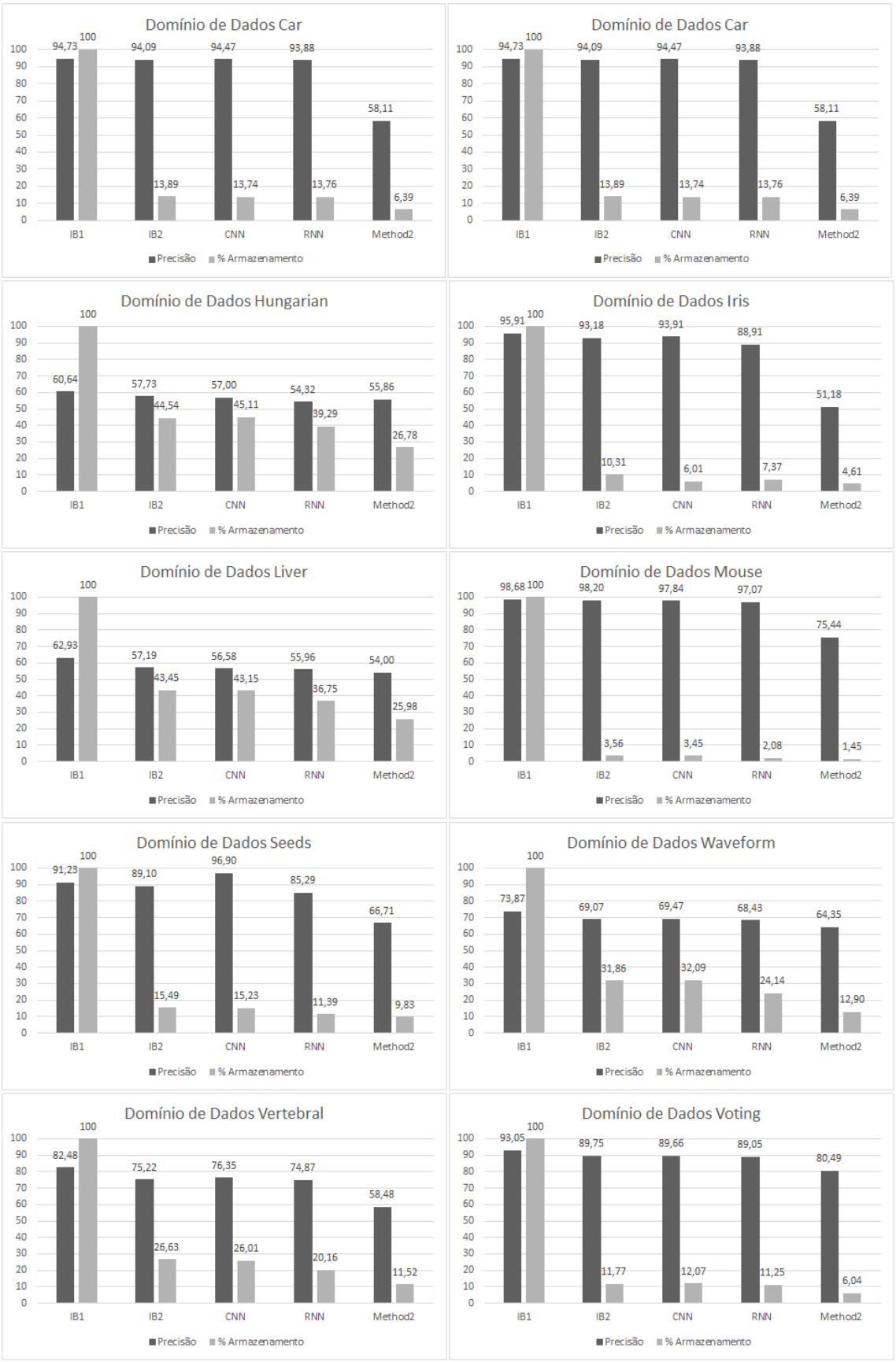


Figura 7.3 Resultados referentes à Precisão e % armazenamento requerido pelos algoritmos de ABI que realizam redução de volume. Para comparação, são apresentados também os resultados relativos do algoritmo IB1, que não realiza redução de volume de dados.

A Figura 7.4 apresenta a média da precisão dos 10 domínios de dados para cada algoritmo utilizado no experimento. Observando simplesmente a média, o Algoritmo IB1 possui a melhor acurácia, porem armazena todo o Conjunto de Treinamento.

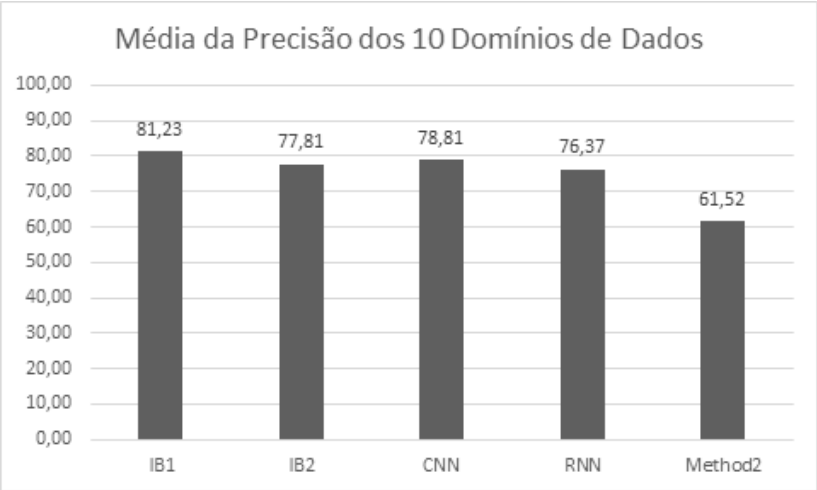


Figura 7.4 Média da Precisão nos 10 Domínios de Dados, para cada algoritmo utilizado no experimento.

Na Figura 7.5 é exibida a média de armazenamento que cada algoritmo armazenou na DC, o algoritmo $Method_2$ possui a menor quantidade de instância armazenadas, mas a média da Precisão é menor que os demais algoritmos do experimento.

Os resultados alcançados evidenciam que os algoritmos IB2 e CNN, possuem um desempenho semelhante, tanto na precisão como na porcentagem de armazenamento. O algoritmo RNN teve melhor desempenho (com relação ao número de instâncias que armazena), que os algoritmos IB2 e CNN para a maioria dos domínios de dados, porem o tempo de processamento para a construção da DC (*Descrição do Conceito*) supera aquele obtido pelos demais algoritmos.

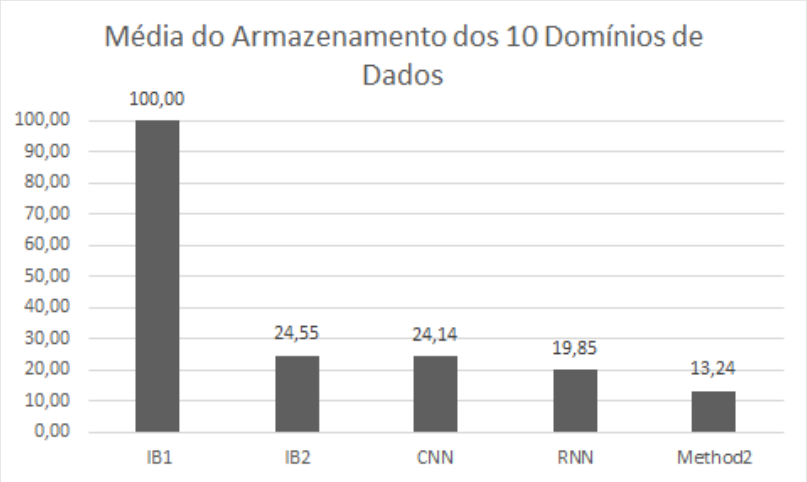


Figura 7.5 Média da Porcentagem do Armazenamento dos 10 Domínios de Dados para cada algoritmo utilizado no experimento.

7.6 Experimentos Complementares, com Técnicas Alternativas para o Tratamento de Valores Ausentes

Para avaliar alternativas de tratamento de valores ausentes, diferentes daquela proposta de [Aha *et al.* 1991] e utilizada nos experimentos descritos na Seção 7.5, foram implementadas no sistema SABI duas alternativas para o tratamento de valores ausentes.

A primeira é referenciada por *Valor Médio dos Valores Mínimos e Máximo dos Atributos Ausentes* e a segunda como *Exclusão da Instância que possua valor ausente de qualquer de seus atributos*.

Como pode ser visto na Tabela 7.1, dentre os conjuntos de dados utilizados nos experimentos, apenas três deles possuem valores ausentes: *Cleveland, Hungarian* e *Voting.* Com vistas a avaliar o impacto de estratégias de imputação de dados nos resultados relacionados à redução de volume de instâncias realizada pelos quatro algoritmos de redução, foram realizados novos experimentos, seguindo os mesmos procedimentos descritos na Seção 7.4. Para o experimento em que ocorre a exclusão das instâncias que possuem valores ausentes, a Tabela 7.5 apresenta a quantidade e a porcentagem de instâncias para os conjuntos de Treinamento e Teste. Como o conjunto de dados *Hungarian* possui uma única instância que não possui valores ausentes, tal conjunto não é considerado para o experimento em que a estratégia de imputação considera a eliminação das instâncias que têm valor de atributo ausente.

Tabela 7.5 DD: domínio de dados; #ID: número de identificação do domínio; #ICO: Número de instâncias no conjunto original; #NVA: valores ausentes de atributos/número de valores ausentes; #NVIA: número de instância com valores ausentes; #ITr: Número de instâncias no conjunto de treinamento; #ITe: Número de instâncias no conjunto de teste.

DD	#ID	#ICO	#NVA	#NIVA	#Itr	#ITe
Cleveland	1	303	6	6	252 (85,15%)	45 (14,85%)
Hungarian	2	294	782	293	-	-
Voting	3	435	392	160	197 (85,06%)	35 (14,94%)

Na Tabela 7.6 e na Figura 7.6 são apresentados os resultados obtidos relativos à precisão e ao percentual de armazenamento, para cada um dos domínios de dados, em que foram utilizadas as técnicas de tratamento de valores ausentes: (1) proposta por [Aha *et al.* 1991], que foram rescritas para ilustrar a comparação, (2) média dos valores mínimo e máximo de cada atributo e (3) exclusão da instância com valores ausentes. Tanto na próxima tabela quanto na próxima figura são apresentados, também, os resultados obtidos com o algoritmo IB1, para comparação.

Tabela 7.6 #DD: domínio de dados; #T Tratamento de Valores Ausentes (1: Aha, 2: Média, 3: Exclusão); Colunas IB1, IB2, CNN, RNN e $Method_2$: são os algoritmos utilizados nos experimentos; #Pc Precisão de cada algoritmo; #% Porcentagem de Armazenamento.

#DD	#T	IB1		IB2		CNN		RNN		$Method_2$	
		#Pc	#%	#Pc	#%	#Pc	#%	#Pc	#%	#Pc	#%
Cleveland	1	58,76	100	54,58	44,03	55,91	44,57	55,87	32,27	50,53	26,88
	2	58,58	100	54,93	44,58	53,42	44,26	56,13	32,22	52,44	27,63
	3	58,84	100	55,73	43,29	55,20	43,38	56,22	36,83	72,47	26,86
Hungarian	1	60,64	100	57,73	44,54	57,00	45,11	54,32	39,29	55,86	26,78
	2	62,86	100	55,82	43,78	56,50	44,46	55,05	37,66	54,27	27,48
	3	-	-	-	-	-	-	-	-	-	-
Voting	1	93,05	100	89,75	11,77	89,66	12,07	89,05	11,25	80,49	06,04
	2	92,89	100	88,52	11,91	88,92	12,49	86,74	09,68	76,34	06,06
	3	90,07	100	87,49	07,45	88,74	07,44	87,49	06,14	80,91	05,06

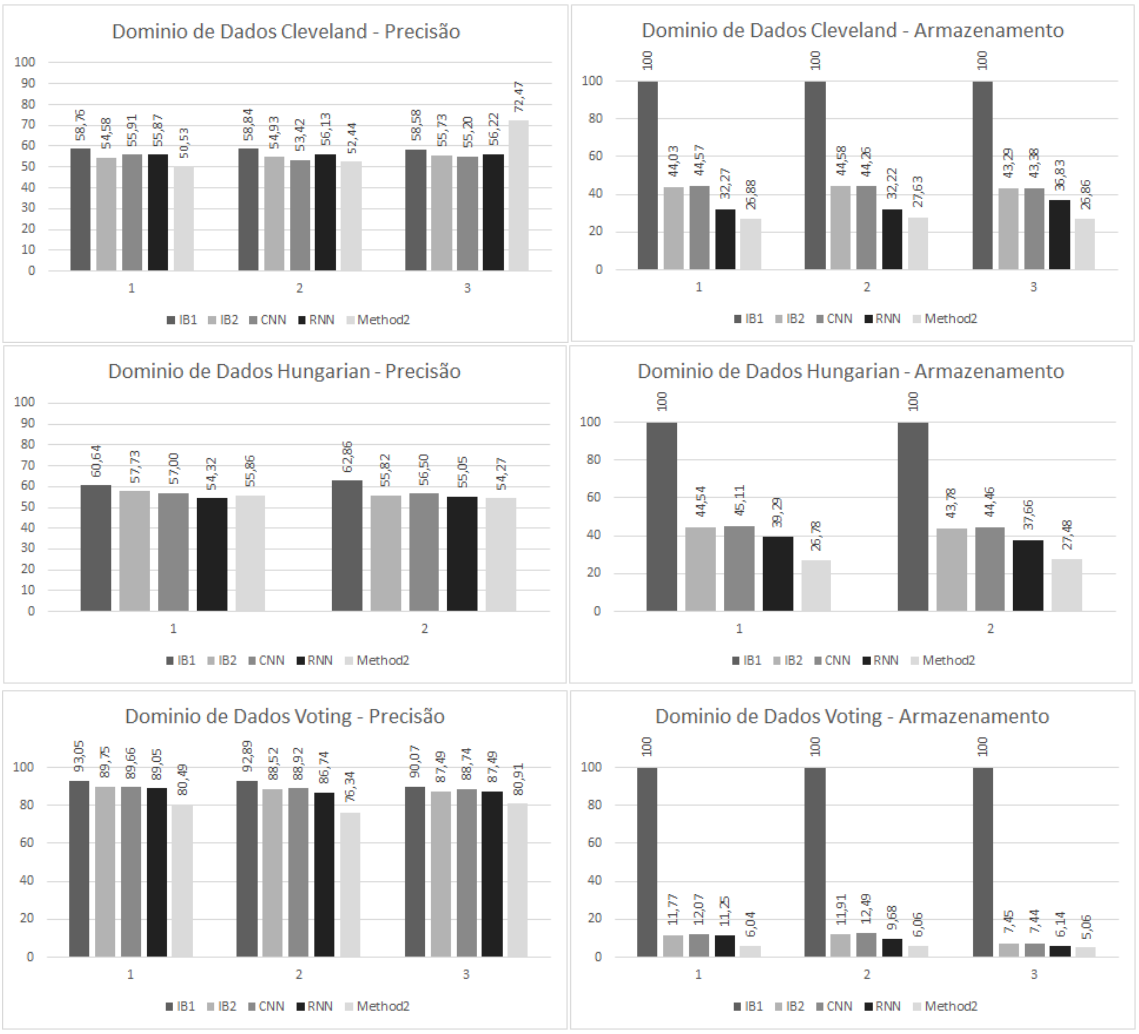


Figura 7.6 Resultado dos Experimentos, utilizando técnicas de tratamento de valores ausentes.

7.7 Discussão Final dos Resultados Obtidos nos Experimentos Realizados

Como esperado, o algoritmo IB1 obteve a melhor acurácia entre os domínios de dados do experimento, considerando que ele não implementa nenhuma técnica de redução de armazenamento, ou seja, todo o conjunto de treinamento é incorporado a DC (*Descrição do Conceito*).

Os algoritmos que implementam redução de volume de instâncias, isto é, o IB2, CNN e RNN, apresentaram resultados estaticamente similares em quase todos os domínios de dados utilizados. Para os domínios *Hungarian*, *Iris* e *Seeds*, entretanto, o algoritmo RNN apresentou ligeira desvantagem com relação à precisão. O algoritmo $Method_2$, apresentou o melhor desempenho apenas para o domínio *Hungarian*, que possui a maior quantidade de valores ausentes.

Comparando os algoritmos IB2, CNN e RNN, os resultados apontaram que o algoritmo RNN produziu a maior redução do volume de instâncias armazenadas em praticamente todos os domínios de dados, com exceção do domínio Iris. Analisando o consumo de processamento o algoritmo RNN é o mais moroso, pois na primeira fase é executado o algoritmo CNN e, apenas após o CNN ter finalizado é que efetivamente o RNN é inicializado.

Em se tratando de volume armazenado de instâncias, o algoritmo $Method_2$ obteve a menor porcentagem de instâncias armazenadas para todos os domínios de dados. Em contrapartida, entretanto, precisão do algoritmo foi prejudicada. Como o *Metthod*$_{2,}$ em sua segunda fase, utiliza o algoritmo CNN, o pré-processamento dos dados de entrada com base nos links *Tomek,* que é realizado durante a primeira fase, talvez seja muito rigoroso para os dominós de dados utilizados nos experimentos.

Para analisar o impacto do tratamento de valores ausentes, além do esquema empregado por Aha e utilizado nos experimentos descritos na Seção 7.5, foram realizados experimentos utilizando dois outros esquemas de imputação, a saber, *Valor Médio dos Valores Mínimos e Máximo dos Atributos Ausentes* e a segunda como *Exclusão da Instância que possua valor ausente de qualquer de seus atributos*. Quando foi aplicada a técnica de valor médio dos valores mínimos e máximo do atributo, houve uma discreta melhora na precisão para o domínio *Cleveland* quando utilizado por todos os algoritmos, com exceção do algoritmo CNN. A porcentagem de armazenamento não sofreu alterações relevantes para este domínio de dados. No domínio de dados *Hungarian* não houve mudanças significativas tanto na precisão como para o armazenamento. Porem para o domínio de dados *Voting* o algoritmo IB1 obteve uma leve melhora na precisão.

Para a técnica de tratamento de valores ausentes que implica a exclusão da instância com o valor ausente, quando o domínio de dados *Cleveland* foi utilizado pelos algoritmos, uma leve melhora tanto na precisão como no armazenamento para quase todos os algoritmos aconteceu, com destaque para o *Method*$_2$ que obteve uma melhora significativa na precisão. O domínio de dados *Hungarian* não foi utilizado neste experimento por conter apenas uma única instância que não continha valor ausente, esse método mostrou-se radical para domínios de dados com grande incidência de valores ausentes. Para o domínio de dados *Voting* apresentou piora na precisão e na redução na quantidade das instâncias armazenadas.

Capítulo 8
Conclusões e Trabalhos Futuros

8.1 Considerações Iniciais

Esse capítulo está organizado em três seções. A Seção 8.2 retoma o objetivo principal da pesquisa que foi o de investigar algoritmos ABI que promovem redução de instâncias armazenadas e, ao mesmo tempo, procuram manter a precisão de classificação alta. Para tanto, apresenta brevemente um resumo dos algoritmos abordados. A Seção 8.3 apresenta as conclusões do trabalho realizado, apontando possíveis linhas de pesquisa que poderão dar continuidade à pesquisa realizada e descrita nesta dissertação.

8.2 Considerações sobre a Pesquisa Conduzida

A pesquisa teve como contexto a área de Aprendizado Indutivo de Máquina e, nela, os Algoritmos Baseados em Instâncias.

Foram evidenciadas e estudadas as principais características de algoritmos baseados em instâncias, bem como nomenclatura e conceituação relacionadas a esse grupo de algoritmos, tendo como foco inicial da pesquisa o algoritmo NN (*Nearest Neighbor-Vizinho Mais Próximo*), considerado o ABI mais popular, porém não necessariamente o mais eficiente, devido à sua natureza de armazenar todo o conjunto treinamento para ser utilizado na classificação de uma nova instância.

Na sequência a família IBL (*Instance-based Learnin*) composta pelos algoritmos IB1, IB2, IB3, IB4 e IB5 foram estudadas. Para o trabalho de pesquisa foram considerados os algoritmos IB1 que é similar ao NN, tendo como diferenciais o fato de processar as instâncias incrementalmente e utilizar uma política de tolerância à ausência de valores de atributos e o algoritmo IB2 que é uma extensão do IB1 que tem por objetivo a redução do volume de dados armazenados, que foi o objetivo desta pesquisa. Apesar dos algoritmos IB3, IB4 e IB5, serem considerados extensão e/ou variação do IB2 não fazem parte deste trabalho, por terem outros focos, que não a redução de armazenamento.

O estudo de ABIs envolveu os principais aspectos de um processo de redução do volume de instancia, (1) nas estratégias de redução do armazenamento, (2) aumento da

velocidade de classificação, (3) tolerância a ruídos, (4) precisão de generalização, (5) exigência de tempo no processo de aprendizado e (6) incrementabilidade.

Um experimento realizado por Aha e descrito em [Aha *et al.* 1991], utilizando o algoritmo IB2, foi reproduzido sob o sistema SABI. A descrição do experimento bem como detalhes relacionados a ele estão no Capitulo 4, em que são discutidos alguns dos conceitos envolvidos e mostrado, via exemplos, os processos envolvidos na indução da descrição do conceito.

Os algoritmos CNN (*Condensed Nearest Neighbour*) [Hart 1968], RNN (*Reduced Nearest Neighbour*) [Gates 1972] e $Method_2$ [Tomek 1976], foram estudados em detalhes (Capitulo 5). O algoritmo SNN (*Selective Nearest Neighbor*) [Ritter1975], que inicialmente fazia parte do projeto de pesquisa, foi substituído pelo algoritmo $Method_2$, devido à ausência de referencial teórico consistente referente ao SNN, que desse suporte para a implementação do algoritmo.

Para a realização dos experimentos foi desenvolvido o sistema SABI (*Sistema para Aprendizado Baseado em Instâncias*), que disponibiliza um ambiente para a pesquisa de algoritmos baseados em instâncias sem redução de volume, assim como de algoritmos baseados em instâncias que implementam redução do conjunto de instâncias. O SABI presentemente disponibiliza implementações dos algoritmos NN, IB1, IB2, CNN, RNN e Method2. O SABI está descrito em detalhes no Capítulo 6 dessa dissertação.

Para avaliar as estratégias de redução de volume de dados armazenados por algoritmos ABI tanto em performance como em armazenamento, foram selecionados 10 domínios de dados, que foram submetidos aos algoritmos pesquisados, sendo que os resultados alcançados foram apresentados e discutidos.

8.3 Conclusões e Continuidade do Trabalho

Com a conclusão do trabalho de pesquisa sobre o desempenho de algoritmos de redução de armazenamento de volume e após a realização dos experimentos descritos no Capítulo 7, é possível constatar que não existe um algoritmo ideal considerando aqueles pesquisados, que obtenha o melhor desempenho para qualquer domínio de dados, tanto em precisão como em armazenamento.

Os domínios de dados que possuem instâncias com valores ausentes ou com ruídos são afetados negativamente tanto em precisão como em armazenamento. Os

experimentos realizados evidenciam que são necessárias técnicas de suavização para tratamento dessas instancias, que sejam eficientes e dinâmicas, e que se adaptem em predizer o melhor valor a ser utilizado para cada caso.

O desempenho do algoritmo CNN tanto em armazenamento como em precisão pode ser destacado, pois para cada domínio de dados de treinamento é construído um subconjunto consistente, mesmo que em muitas situações o subconjunto gerado não seja um subconjunto mínimo.

Existem diversos algoritmos que implementam redução do volume de instâncias, esta dissertação limitou-se em algoritmos baseado no NN. Como trabalhos futuros, pesquisas podem ser iniciadas em algoritmos de redução de volume de armazenamento que implementem técnicas que promovam a redução de ruídos nos dados, tais como IB3, DROP2-DROP5 e DEL discutidos em [Wilson & Martinez 2000].

O sistema computacional desenvolvido com o propósito de disponibilizar um ambiente de ABI para testes e validações, o SABI, por ser modular, pode ser facilmente estendido com o acoplamento de implementações de novos algoritmos de redução de volume de armazenamento. Novas técnicas para a suavização de valores ausentes ou técnicas de tratamento que identifiquem valores com ruídos, podem ser implementadas e acopladas ao sistema SABI.

Referências Bibliográficas

[Aha *et al.* 1991] Aha, D. W.; Kibler, D.; Albert, M. K. (1991) Instance-based learning algorithms, *Machine Learning*, v. 6, pp. 37–66.

[Aha 1992] Aha, D. W. (1992) Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms, *International Journal of Man-Machine Studies*, 36, 267–287.

[Aha 2013] Aha, D. W. (Ed.) (2013) Lazy Learning, Springer Science+Business Media Dordrecht.

[Angiulli 2005] Angiulli, F. (2005) Fast condensed nearest neighbor rule, *Machine Learning*, pp. 25-32.

[Bhatia & Vandana 2010] Bhatia, N.; Vandana (2010) Survey of Nearest Neighbor Tecniques, IJCSIS Intr. Jour. Of Comp. Sci. and Inf. Sec., Vol. 8, no. 2.

[Bishop 2006] Bishop, C. M. (2006) Pattern Recognition and Machine Learning, Springer Science+Business Media, LLC.

[Breiman *et al.* 1984] Breiman, L.; Friedman, J.; Stone, C. J.; Olshen, R. A. (1984) Classification and regression trees, CRC Press.

[Chang 1974] Chang, C.-L. (1974) Finding prototypes for nearest neighbor classifiers, *IEEE Transactions on Computers*, v. 23, no. 11, pp. 1179–1184.

[Clark & Niblett 1989] Clark, P.; Niblett, T. (1989) The CN2 induction algorithm, *Machine Learning*, v. 3, no. 4, pp. 261-283.

[Cover & Hart 1967] Cover, T.M.; Hart, P.E. (1967) Nearest neighbour pattern classification, *IEEE Transactions on Information Theory*, v. 13, pp. 21-27.

[Cover & Hart 1967] Cover, T. M.; Hart, P. E. (1967) Nearest neighbor pattern classification, *IEEE Transactions on Information Theory*, v. 13, no. 1, pp. 21–27.

[Dasarathy 1991] Dasarathy, B. V. (1991) Nearest neighbor (NN) norms: NN pattern classification techniques. Los Alamitos, CA: IEEE Computer Society Press.

[Domingos 1995] Domingos, P. (1995) Rule induction and instance-based learning: a unified approach, In: Proc. of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal, Canada, pp. 1226–1232.

[Domingos 1996] Domingos, P. (1996). Unifying instance-based and rule-based induction, *Machine Learning*, v. 24, pp. 141–168.

[Duda et al. 2001] Duda, R. O.; Hart, P. F.; Stork, D. G. (2001) *Pattern Classification*, USA: John Wiley & Sons, Inc.

[Dudani 1976] Dudani, S. A. (1976) The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*, v. 6, no. 4, pp. 325–327.

[Galiardi 2011] Gagliardi, F (2011) Instance-based classifiers applied to medical databases: diagnosis and knowledge extraction, *Artificial Intelligence in Medicine*, v. 52, no. 3, pp. 123–139.

[Gates 1972] Gates, G.W. (1972) The reduced nearest neighbor rule, *IEEE Transactions on Information Theory*, v. 18, no. 3, pp. 431–433.

[Hart 1968] Hart, P. E. (1968) The condensed nearest neighbor rule, *IEEE Transactions on Information Theory*, v. 14, pp. 515–516.

[Jain 1988] Jain, A.K.; Dubes, R.C. (1988) *Algorithms for Clustering Data*, Prentice Hall.

[Kibler & Aha 1987] Kibler, D. & Aha, D. W. (1987) Learning representative exemplars of concepts: an initial case study, In: Proceedings of the Fourth International Workshop on Machine Learning, Irvine, CA: Morgan Kaufmann, pp. 24–30.

[Lichman 2013] Lichman, M. (2013) UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[Mitchell 1997] Mitchell, T. M (1997) *Machine Learning*, USA: McGraw-Hill.

[Nietto & Nicoletti 2017] Nietto, P.; Nicoletti, M. C. (2017). Case studies in divisive hierarchical clustering. *International Journal of Innovative Computing and Applications*. v. 8, pp. 102–112.

[Nicoletti 1994] Nicoletti, M. C. (1994) Ampliando os limites do aprendizado indutivo de máquina através das abordagens construtiva e relacional, Ph. D., IFSC-USP.

[Quinlan 1993] Quinlan., J. R. (1993) C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers.

[Ritter *et al.* 1975] Ritter, G. L.; Woodruff, H. B.; Lowry, S. R.; Isenhour, T. L. (1975) An algorithm for a selective nearest neighbor decision rule, *IEEE Transactions on Information Theory*, v. 21, no. 6, pp. 665–669.

[Salzberg 1991] Salzberg, S. (1991) A nearest hyperrectangle learning method, *Machine Learning*, v. 6, pp. 251–276.

[Santos & Nicoletti 1996] Santos, F.O.; Nicoletti, M.C. (1997) O modelo de aprendizado baseado em instâncias-algoritmo IB1, RT-DC 008/96, UFSCar/DC, São Carlos, 19 pg.

[Santos & Nicoletti 1997] Santos, F.O.; Nicoletti, M.C. (1997) A família de algoritmos instance-based learning (IBL), RT-DC 006/97, UFSCar/DC, São Carlos, SP, 26 pg.

[Sproull 1991] Sproull, R. F. (1991) Refinements to nearest-neighbor searching in k-dimensional trees. Algorithmica, v. 6, pp. 579–589.

[Theodoridis & Koutroumbas 1999] Theodoridis, S.; Koutroumbas, K. (1999) *Pattern Recognition*, USA: Academic Press.

[Tomek 1976] Tomek, I. (1976) An experiment with the edited nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 6, no. 6, pp. 448–452.

[Wettschereck 1994] Wettschereck, D. (1994) A hybrid nearest-neighbor and nearest-hyperrectangle algorithm, In F. Bergadano & Raedt, L. de (Eds.), In: Proceedings of the 7th European Conference on Machine Learning, LNAI, v. 784, pp. 323–335.

[Wettschereck & Dietterich 1995] Wettschereck, D. & Dietterich, T. G. (1995) An experimental comparison of nearest-neighbor and nearest hyperrectangle algorithms, *Machine Learning*, v. 19, no. 1, pp. 5–28.

[Wettschereck *et al.* 1997] Wettschereck, D., Aha, D.W. & Mohri, T. (1997) A review and comparative evaluation of feature weighting methods for a class of lazy learning algorithms, *Artificial Intelligence Review*, 11 (Special issue on Lazy Learning), pp. 273–314.

[Wilson & Martinez 1997a] Wilson, D. R.; Martinez, T. R. (1997a) Improved heterogeneous distance functions, *Journal of Artificial Intelligence Research (JAIR)*, v. 6, no. 1, pp. 1–34.

[Wilson & Martinez 1997b] Wilson, D. R.; Martinez, T. R. (1997b) Instance pruning techniques. In: Fisher, D. (Ed.), Machine Learning: Proceedings of the Fourteenth International Conference (ICML'97), San Francisco, CA: Morgan Kaufmann Publishers, pp. 403–411.

[Wilson & Martinez 2000] Wilson, D. R.; Martinez, T. R. (2000) Reduction techniques for instance-based learning algorithms, *Machine Learning*, v. 38, pp. 257–286.

[Wu et al. 2008] Wu, X.; Kumar, Y.; Quinlan,· J. R.; Ghosh, J.; Yang , Q.; Motoda, H.; McLachlan, G. J.; Ng, Angus; Liu, B.; Yu, P. S.; Zhou, Z.-H.; Steinbach, M., Hand, D. J.; Steinberg, D. (2008) Top 10 algorithms in data mining, *Knowledge Information System*, v. 14, pp. 1-13.

[Zhang 1992] Zhang, J. (1992) Selecting typical instances in instance-based learning, In: Proceedings of the Ninth International Conference on Machine Learning, Aberdeen, Scotland, pp. 470–479.

Anexo

Submissão do Artigo Reducing Data Volume in Instance Based Learning para o European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2018).

Maria do Carmo Nicoletti Logout

ESANN
Home
Call for papers
Special sessions
Author guidelines
Submission
Program
Infos speakers
Venue
Social events
Bruges
Registration
Hotels
Committees
Sponsors
Photos
Proceedings
Past ESANN
Get a password
About

MyESANN
Personal info
My submissions
My registrations

My submissions

Previous ESANN2018-20 Next

Details | Reviews

Owner(s) [Add owner]

The owners of a paper are authorized to modify the submission. If you want to share this right with your co-authors, please edit this field (you must know their login).

- Maria do Carmo Nicoletti

Title [Edit]

Please do not capitalize words (except the first one of the title), unless you would capitalize them in the core of the paper too.

reducing data volume in instance based learning

Abstract [Edit]

During the training phase of the Nearest-Neighbor (NN) algorithm, considered the most popular Instance-Based Learning (IBL) algorithm, all training instances are simply stored, and they represent the description of the learned concept. IBL algorithms postpone the generalization process, which usually happens during the training phase, until the classification phase starts i.e., when an unclassified data instance needs to be classified. When training sets have a high volume of instances, to store them all becomes unfeasible mainly due to storage requirements. Several IBL algorithms overcome storage related problems by implementing data volume reduction i.e., by storing only a representative subset of the whole training set. The investigation described in this paper focuses on four IBL algorithms that implement data reduction, which have been empirically evaluated in data sets from the UCI Repository. Results of their performance, considering storage reduction and classification accuracy, are presented and discussed.

Keywords [Edit]

Please provide keywords that can help to assign reviewers to your paper.

machine learning;instance-based learning;volume reduction

Authors [Add author]

It is important to list all authors here, in the same order as mentioned in the paper. Any missing or wrong information can lead to errors in the conference programme and proceedings. Use [+] and [-] to move up and down respectively the names in the list, [X] to delete and [C] to identify the corresponding author (mandatory).

Warning: it is not possible for an author to change his/her details (affiliation, address, swapping between first name and name) once the details have been entered into the author database. If you need to change your details, please send an e-mail to esann@dice.ucl.ac.be.

[+] [-] [X] [C] Nicoletti Maria do Carmo (Corresponding author)
carmo@dc.ufscar.br

[+] [-] [X] [C] Claudiano Luis Andre
laclaudi@uol.com.br

Session [Edit]

My paper is to be submitted to the following session:

Regular sessions

Type of presentation [Edit]

https://www.elen.ucl.ac.be/esann/index.php?pg=mysubmissions 1/2

Preferred type of presentation:

oral preferred

Files [Add file]

Please upload your paper according to the format detailed in the instructions to authors.
Warning when downloading papers: depending on your browser, it may happen that you have first to save the paper after download, and then open the saved version. Opening it directly without saving may fail and produce an error message.

	Name	Date	Time
[X]	Nicoletti_Claudiano_ESANN_2018VFF.pdf (Submission)	3/11/2017	12:47