A Família de Algoritmos *Instance-Based Learning* (IBL)

Flávia Oliveira Santos

Maria do Carmo Nicoletti

Universidade Federal de São Carlos (UFSCar)

Departamento de Computação (DC)

C.P. 676 — 13565-905 — São Carlos - SP - Brasil

***e-mail* : {flavia,carmo} @dc.ufscar.br**

Resumo. Este relatório apresenta e discute os algoritmos IB2, IB3, IB4 e IB5, representantes da família IBL (*Instance-Based Learning*). Esta família é considerada uma família de algoritmos descendentes do algoritmo *Nearest Neighbour* (NN), cuja técnica principal se baseia em classificar uma nova instância considerando-se a classe de sua vizinha mais próxima. A família IBL é formada por diversas variações do algoritmo NN, cada uma delas proposta com o objetivo de contornar alguma limitação ou do NN ou de uma sua versão prévia. À medida que os algoritmos dessa família se sofisticam (novas versões são propostas), mecanismos para contornar alguns dos problemas evidenciados em domínios reais são incorporados, tais como o tratamento de ruídos nos dados, o tratamento de atributos irrelevantes e a introdução de novos atributos.Este relatório é para ser lido como uma continuidade do relatório "O Modelo de Aprendizado Baseado em Instâncias - Algoritmo IB1" (RT-DC 008/96), onde são apresentadas as principais idéias que subsidiam o modelo de aprendizado baseado em instâncias. Nele é também apresentado o algoritmo IB1, o primeiro representante da família IBL, e é feita a sua análise sob a perspectiva de aprendizado pac.

Palavras-chaves: aprendizado de máquina, aprendizado baseado em instâncias, família IBL.

1 Introdução

Os algoritmos de aprendizado da família IBL armazenam instâncias de treinamento na memória como pontos no espaço n-dimensional, definido pelos n atributos que as descrevem, e nunca mudam a representação desses pontos. A expressão do conceito é, pois, representada por aquele conjunto de pontos. As duas decisões mais relevantes a serem tomadas são: quais pontos armazenar e qual métrica adotar, quando da fase de classificação para *medir* a distância de uma nova instância às instâncias que fazem parte da expressão do conceito. Segundo Aha et al. [Aha 91, Aha 92], as diversas variantes do modelo *baseado em instâncias*, representadas pelos algoritmos da família IBL, objetivam descobrir os limites dessa abordagem de aprendizado que armazena instâncias de treinamento sem realizar qualquer tipo de generalização sobre elas. Todas essas variantes usam a técnica de "vizinho mais próximo" para classificar novos exemplos. Os algoritmos tratados nesse documento foram propostos por D. W. Aha e encontram-se descritos em [Aha 91, Aha 92].

A família IBL pode ser considerada como uma extensão do algoritmo NN [Cover 67]. Os algoritmos da família IBL procuram contornar algumas das sérias limitações associadas ao modelo NN que, segundo [Breiman 84], tem como inconvenientes :

- serem classificadores computacionalmente caros uma vez que armazenam todas as instâncias de treinamento;
- serem 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;
- serem intolerantes a atributos com ruídos e a atributos irrelevantes;
- fornecerem pouca informação útil com relação à estrutura dos dados.

Uma descrição do conceito baseada em instâncias inclui um conjunto de instâncias armazenadas e, possivelmente, alguma informação com relação aos seus desempenhos anteriores, durante o processo de classificação. Esse conjunto de instâncias pode mudar após cada instância de treinamento ter sido processada.

Para os algoritmos da família IBL a descrição do conceito não se limita apenas ao conjunto das instâncias armazenadas; inclui também as funções de *similaridade* e *classificação*, bem como a maneira que essas funções usam o conjunto das instâncias armazenadas. Essas funções determinam como o conjunto de instâncias, que é parte da descrição do conceito, é usado para predizer a classe de novas instâncias, ou seja, como tal conjunto é usado durante a fase de classificação. Conseqüentemente, os três componentes que caracterizam os algoritmos da família IBL são :

- Função de Similaridade — responsável pelo cálculo da similaridade entre uma instância de treinamento *i* e as instâncias que participam da descrição do conceito.
- Função de Classificação — responsável pela classificação da instância *i*. Para fornecer a classificação de *i*, usa tanto o resultado da função de similaridade quanto os registros de desempenho de classificações anteriores das instâncias que descrevem o conceito.
- Atualizador da Descrição do Conceito — responsável pelos registros de desempenho de classificação e pela decisão de qual instância incluir na descrição do conceito. Recebe como entrada a instância *i*, os resultados da função de classificação, bem como a descrição atual do conceito. É o responsável pela modificação da descrição do conceito.

Os algoritmos IBL assumem que instâncias similares têm classificações similares. Isso implica no uso de uma heurística local para a classificação de novas instâncias, de acordo com a classificação do vizinho “mais semelhante”. Este trabalho apresenta e discute a família IBL

e está organizado como se segue. Na Seção 2 o algoritmo NN é brevemente apresentado, objetivando contextualizar a técnica do "vizinho mais próximo" e fornecer subsídios para a família IBL. Nas Seções de 3 a 6 são descritos os funcionamentos dos algoritmos IB2, IB3, IB4 e IB5, respectivamente, e os tratamentos incorporados por esses algoritmos na solução de alguns dos inconvenientes do modelo NN apontados anteriormente. A Seção 7 apresenta um resumo das principais características dos algoritmos da família IBL identificadas neste relatório e a Seção 8, as conclusões desse trabalho.

2 *Nearest Neighbour* (NN)

Recentemente, tem-se notado um crescente investimento em técnicas que, assim como o IBL, têm como base o algoritmo Nearest Neighbor (NN) devido, principalmente, à sua simplicidade e habilidade em produzir classificações com alto grau de precisão após um rápido aprendizado. O algoritmo (ou regra) NN pode ser encontrado em várias referências na literatura [Beale 94, Gates 72, Hart 68]. Como é o algoritmo no qual se fundamenta a família IBL, ele é tratado brevemente a seguir.

Considere um espaço bidimensional (definido pelos atributos a_1 e a_2) onde a classe 1 e a classe 2 estão representadas por 10 e 12 instâncias, respectivamente, e é necessário determinar a qual das duas classes pertence uma instância x, de classe desconhecida. Esta situação é ilustrada na Figura 2.1.

Basicamente, as técnicas NN assumem como classe da instância x, a classe da instância que está mais próxima de x; no exemplo da Figura 2.1, x assume a classe 2. A Tabela 2.1 apresenta a definição formal da técnica NN, baseada na definição proposta em [Gates 72].

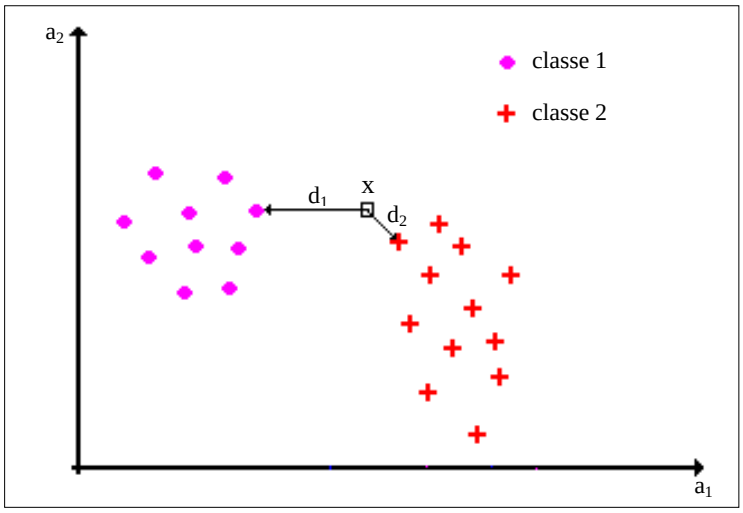

Figura 2.1. Processo de classificação realizado pelo algoritmo NN onde x: instância de classe desconhecida, d_1: menor distância à classe 1 e d_2: menor distância à classe 2

Sejam:

- espaço n-dimensional de atributos
- M classes, numeradas 1, 2, ..., M
- p instâncias de treinamento, cada uma expressa como um par (x_i, θ_i), para $1 \leq i \leq p$, onde:
  a) x_i é uma instância de treinamento, expressa por um vetor de pares *atributo-valor*
  $$x_i = (x_{i_1}, x_{i_2}, ..., x_{i_n})$$
  b) $\theta_i \in \{1,2,...,M\}$ denota a classe correta da instância x_i

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

$$d(x,x_j) \leq d(x,x_i) \qquad 1 \leq i \leq p$$

onde d é alguma métrica n-dimensional de distância.

Tabela 2.1. Definição formal da técnica NN

A regra $d(x,x_j) \leq d(x,x_i)$ é mais apropriadamente chamada de regra 1-NN, uma vez que usa apenas um "vizinho mais próximo", i.e., calcula a distância da nova instância a cada ***um*** dos exemplos do conjunto de treinamento. Uma das variantes da regra 1-NN é conhecida como k-NN, a qual identifica as k instâncias mais próximas $\{i_1, i_2, ..., i_k\}$ e decide pela escolha da classe que comparece com maior freqüência no conjunto $\{\theta_{i_1}, \theta_{i_2}, ..., \theta_{i_k}\}$. Esta regra é vista como um método para o tratamento de ruídos pois tenta prevenir que uma instância com ruído classifique incorretamente uma nova instância. Para determinar o valor de k um método

bastante utilizado é o *cross validation*, onde o sistema é testado usando-se vários valores para k e aquele valor que produzir o melhor resultado para aquela aplicação é adotado posteriormente. Quanto maior a quantidade de ruído nos dados, maior deve ser o valor de k.

A variante do NN, conhecida como CNN (Condensed Nearest Neighbor) [Hart 68], continua ainda escolhendo a classe do vizinho mais próximo. Entretanto, opta pela escolha de um subconjunto de T_{NN}, denominado T_{CNN}, que tem um desempenho quase tão bom quanto o T_{NN} na classificação de novas instâncias. No caso específico do T_{CNN}, como comentado em [Gates 72] (pág. 431), "a eventual queda no desempenho, devido à escolha de um conjunto bem menor de pontos, é compensada pela redução de memória necessária para o armazenamento do conjunto de treinamento e pelo menor tempo gasto na classificação de uma nova instância".

Na construção do T_{CNN} é empregada a noção de subconjunto consistente, definido como um subconjunto de T_{NN} que classifica todas as instâncias de T_{NN}. O subconjunto minimal é o menor dos subconjuntos consistentes e, conseqüentemente, aquele que irá classificar, mais eficientemente e apropriadamente, toda a instância presente em T_{NN}. O algoritmo T_{CNN} é consistente, mas não é garantido ser minimal. Segue a sua descrição em pseudocódigo na Tabela 2.2 e um exemplo de aplicação, encontrados em [Gates 72].

```
T_CNN ← primeira instância de T_NN
INSTÂNCIA ← primeira instância de T_NN
while todas as instâncias de T_NN não forem corretamente classificadas por T_CNN do
    if T_CNN classifica INSTÂNCIA
        then INSTÂNCIA ← próxima instância de T_NN
        else
            begin
                T_CNN ← T_CNN ∪ {INSTÂNCIA}
                INSTÂNCIA ← primeira instância de T_NN
            end
end-while
```

Tabela 2.2. Algoritmo CNN. O conceito é representado pelos pontos em T_{CNN}

Como pode ser visto na Figura 2.2, o uso do algoritmo CNN no conjunto de treinamento

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)}

onde cada instância é descrita por um atributo e uma classe associada, usando a notação (valor_do_atributo,classe), evidencia o conjunto reduzido

$$T_{CNN} = \{(-13,1),(-6,2),(13,1),(4,2)\}$$

como expressão do conceito.

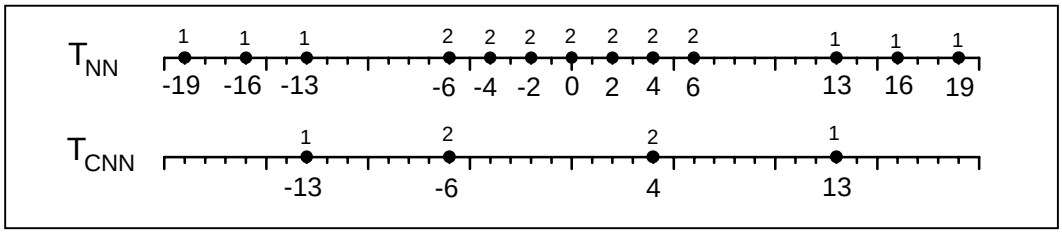

Figura 2.2. Exemplo de uso do algoritmo T_{CNN}

3 Algoritmo IB2

Antes da apresentação do IB2 é importante que alguns detalhes do IB1 sejam apresentados. O algoritmo IB1 armazena todas as instâncias de treinamento, que são processadas incrementalmente. A função de similaridade usada pelo IB1 é definida como mostra a Tabela 3.1.

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

onde
- n : quantidade de atributos usados para descrever as instâncias
- $f(x_i, y_i) = (x_i - y_i)^2$, para atributos numéricos
- $f(x_i, y_i) = (x_i \neq y_i)$, para atributos com valores simbólicos e/ou booleanos

Tabela 3.1. Função de similaridade adotada pelo algoritmo IB1

A função similaridade(x,y) é baseada na distância Euclidiana [Boulos 87] (pág. 219). De fato o cálculo de $\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$ encontra um valor que define a distância entre dois pontos (x e y) no espaço n-dimensional. O ponto y mais próximo de x será aquele com a menor distância. Já a função similaridade(x,y), adotada pelos algoritmos da família IBL, visa obter a instância y mais similar à instância x. No espaço euclidiano onde a métrica da similaridade

adotada é a da proximidade, medida através da distância euclidiana, a menor distância deve representar a maior similaridade, razão pela qual a similaridade é definida como a distância negativa. Por exemplo, considerando um espaço tridimensional, sejam y = (y_1=3, y_2=4, y_3=2) e z = (z_1=3, z_2=2, z_3=6) duas instâncias armazenadas. Na Tabela 3.2 estão calculadas as similaridades entre uma nova instância x = (x_1=10, x_2=4, x_3=2) e cada uma das instâncias y e z, usando a função similaridade definida na Tabela 3.1.

$$\text{similaridade}(x,y) = -\sqrt{\sum_{i=1}^{3} f(x_i, y_i)} = -\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2} =$$

$$= -\sqrt{(10-3)^2 + (4-4)^2 + (2-2)^2} = -\sqrt{49} \Rightarrow \boxed{\text{similaridade}(x,y) = -7}$$

$$\text{similaridade}(x,z) = -\sqrt{\sum_{i=1}^{3} f(x_i, z_i)} = -\sqrt{(x_1 - z_1)^2 + (x_2 - z_2)^2 + (x_3 - z_3)^2} =$$

$$= -\sqrt{(10-3)^2 + (4-2)^2 + (2-6)^2} = -\sqrt{69} \Rightarrow \boxed{\text{similaridade}(x,y) = -8.31}$$

Tabela 3.2. Similaridades entre a nova instância x e cada uma das instâncias y e z

Desta forma, o cálculo da similaridade(x,y) e similaridade(x,z) mostra que a instância x está mais próxima da instância y, pois similaridade(x,y) > similaridade(x,z).

No IB1, se o valor de um determinado atributo está ausente, o algoritmo assume como esse valor, o valor mais diferente possível do valor corrente. Se a função de similaridade está sendo calculada entre

$$y = (y_1 = v_{j_1},\ y_2 = v_{j_2}, \ldots, y_n = v_{j_n}) \qquad \text{e} \qquad x = (x_1 = v_{i_1},\ x_2 = ?, \ldots, x_n = v_{i_n}),$$

o valor do atributo x_2 será assumido como aquele mais diferente possível de v_{j_2}. Se, entretanto, ambos os valores, tanto na representação de y quanto na de x, estiverem ausentes, $f(x_i,y_i)=1$. Este método é usado também nas versões posteriores, IB2, IB3 e IB4. Deve-se ressaltar, entretanto, que o uso de tal técnica não é uma política muito eficiente. Essa abordagem de assumir um determinado valor para o valor ausente, na maioria das vezes pode

estar acarretando uma modificação no conceito, principalmente em situações em que instâncias têm muitos valores ausentes. O algoritmo, ao assumir um valor para cada um dos valores ausentes, pode estar introduzindo um ruído nos dados que pode tornar o seu comportamento tendencioso. O que parece ser uma melhor forma de tratar atributos com valores ausentes é a técnica empregada pelo IB5 que apenas ignora atributos que têm valores ausentes. Não existe a tentativa, no IB5, de assumir um valor para aqueles ausentes. O motivo dessa versão não adotar a mesma técnica será explicado posteriormente, onde o algoritmo IB5 é detalhado separadamente.

O IB2, uma variante do IB1, tem as mesmas funções de similaridade e classificação do IB1; a diferença entre os dois está no atualizador da descrição do conceito que, no IB2, armazena apenas as instâncias classificadas incorretamente, como pode ser visto em sua descrição na Tabela 3.3. A incorporação dessa estratégia de armazenamento foi subsidiada 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 nenhuma instância da descrição do conceito depois que ela foi armazenada.

```
DC ← ∅
for_each x∈ Conjunto de Treinamento do
    begin
        1. for_each y∈ DC do
           sim[y] ← similaridade(x, y)
        2. y_max ← algum y∈ DC  com a maior sim[y]
        3. if classe(x) = classe(y_max)
           then classificação ← correta
           else
              begin
                  3.1. classificação ← incorreta
                  3.2. DC ← DC ∪ {x}
              end
    end
```

Tabela 3.3. Algoritmo IB2 (DC = Descrição do Conceito)

Apesar do IB2 diminuir a necessidade de armazenamento, ele é muito mais sensível à presença de ruído no conjunto de treinamento do que o IB1, uma vez que salva apenas

instâncias com classificação incorreta. A sensibilidade a ruídos é uma conseqüência do fato do IB2, durante a fase de aprendizado, apenas incorporar à expressão do conceito aquelas instâncias que são classificadas incorretamente; instâncias com ruídos são, quase sempre, classificadas erradamente e, conseqüentemente, passam também a descrever o conceito. Em domínios com alta incidência de ruídos, tais instâncias participam pesadamente da expressão final do conceito, afetando de forma negativa o desempenho do sistema quando da fase de classificação, como mostram os experimentos conduzidos por Aha et alii, descritos em [Aha 92] e comentados a seguir. Por outro lado, em domínios sem a presença de ruídos, o IB2 reduz significativamente a necessidade de armazenamento requerida pelo IB1, e os dois algoritmos têm precisão de classificação similares.

Os experimentos, descritos em [Aha 91], que buscaram evidenciar "até que ponto" o IB2 sacrificaria a precisão de classificação para obter um baixo nível de armazenamento, tiveram, para os domínios descritos na Tabela 3.4, os resultados descritos na Tabela 3.5.

Domínios de dados[1]	**Tamanho do conjunto de treinamento**	**Tamanho do conjunto de teste**	**Número de atributos**	**Número de classes**
Voting	350	85	16	2
Primary Tumor	275	64	17	22
LED Display	200	500	7	10
Waveform	300	500	21	3
Cleveland	250	53	13	2
Hungarian	250	44	13	2

Tabela 3.4. Características dos domínios de dados

Domínios de dados	**IB1**		**IB2**	
Voting	91.8 ± 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 3.5. Percentual de precisão ± desvio padrão e percentual de armazenamento

[1] São usados os nomes originais dos domínios constantes do UCI Repositório de dados [Murphy 95], uma vez que fica mais fácil sua identificação e obtenção

Os experimentos realizados permitiram concluir que, muito embora o IB2 reduza, às vezes significativamente, o número de instâncias armazenadas, ele também sacrifica a precisão de classificação, principalmente quando os domínios têm ruídos ou têm instâncias "excepcionais[2]". Algumas das razões apontadas pelos autores para justificar os valores da Tabela 3.5 são descritas nas Tabelas 3.6 e 3.7.

Domínios de dados	Características dos domínios de dados que influenciaram o aprendizado
Voting	• conjunto de dados sem ruídos • classes linearmente separáveis
Primary Tumor	• características do conjunto de dados são variadas • descrição esparsa do conceito: entre os 22 conceitos descritos neste conjunto, existem 11 descritos por menos que 10 instâncias (cada um deles)
Led Display e Waveform	• domínios artificiais com grande quantidade de ruído
Cleveland e Hungarian	• conjunto com dados reais de diagnósticos médicos • domínios com conceitos não muito bem definidos; atributos não caracterizam completamente o conceito

Tabela 3.6. Características dos conjuntos de dados que influenciaram os resultados

Domínios de dados	Análise dos resultados obtidos
Voting	• IB1 e IB2 tiveram precisão de classificação semelhantes. O IB2, entretanto, exigiu muito menos "armazenamento" que o IB1 • esse mesmo comportamento foi observado com vários outros conjuntos de dados relativos a domínios com características similares
Primary Tumor	• a economia de armazenamento não é muita devido ao fato do conjunto de dados ser esparso: não existem instâncias em número suficiente, em cada conceito, para determinar precisamente onde estão os seus limites
Led Display e Waveform	• a precisão de classificação do IB2 foi substancialmente inferior à do IB1, em ambos os domínios. Isso ocorre porque o IB2 classificou erradamente e, subseqüentemente, salvou a maioria das instâncias com ruído. As instâncias com ruído armazenadas foram usadas para classificar incorretamente as próximas instâncias de treinamento. Esse desempenho motivou a proposta do IB3, uma extensão do IB2 tolerante a ruído, abordada na próxima seção
Cleveland e Hungarian	• muito embora o IB2 tenha reduzido significativamente o volume de instâncias armazenadas, em ambos conjuntos de dados, ele teve sua precisão de classificação prejudicada devido aos atributos não descreverem completamente o conceito. Esse fato provocou o armazenamento de um grande número de instâncias consideradas "excepcionais". Essas instâncias são muito semelhantes a instâncias que têm ruídos, uma vez que classificam "pobremente" instâncias similares. Conseqüentemente, o IB2 tem um desempenho ruim nesses tipos de conjuntos de dados, pelas mesmas razões apontadas em domínios com ruídos

Tabela 3.7. Análise dos resultados obtidos para o algoritmos IB2

A característica do IB2, de eleger principalmente as instâncias que estão próximas à fronteira do conceito, como as que o descrevem, pode também ser observada com relação ao algoritmo CNN, conforme comentário do seu autor em [Hart 68]. O CNN, descrito na Seção 2 — Tabela 2.2 (pág. 6), difere do IB2 apenas por :

[2] Caracterizadas por atributos que não as descrevem apropriadamente

- não normalizar os valores de atributo
- ser iterativo com relação ao conjunto de treinamento

4 Algoritmo IB3

O IB3 é um outro algoritmo da família IBL que se propõe a contornar o problema empiricamente evidenciado com o IB2, aquele da sensibilidade a ruídos[3] nos dados. Ruídos em dados, geralmente, implicam maior número de instâncias armazenadas e menor precisão de classificação. O IB3 "filtra" as instâncias armazenadas de maneira a neutralizar a participação, nas decisões de classificação, das prováveis instâncias com ruído.

O IB3 assume que instâncias com ruído terão precisão de classificação pobre devido a seus vizinhos similares, no espaço de instâncias, invariavelmente terem outras classificações; instâncias com ruído são detectáveis porque têm precisão de classificação relativamente baixa [Aha 92]. Com o objetivo de tornar as instâncias com ruídos detectáveis devido à sua precisão de classificação relativamente baixa, o IB3 :

- mantém registros de classificação de todas as instâncias armazenadas, i.e., o número de tentativas de classificação corretas e incorretas. O registro de classificação de uma instância exibe o seu desempenho de classificação em instâncias de treinamento subseqüentemente apresentadas
- usa um teste de significância para determinar quais instâncias são boas classificadoras e quais não são (candidatas a serem instâncias com ruídos). As instâncias consideradas "boas" continuam participando da descrição do conceito e são usadas para classificar novas instâncias apresentadas subseqüentemente. Aquelas consideradas "não boas" classificadoras são descartadas da descrição do conceito

[3] Definido como um valor incorreto de atributo

O IB3, durante a fase de treinamento, usa um teste baseado em intervalos de confiança com o objetivo de decidir se “incorpora” ou não uma instância de treinamento à expressão do conceito. São usados dois intervalos de confiança para cada instância, o *intervalo de precisão de classificação da instância* (IP) e o *intervalo de freqüência da classe observada* (IF). Intervalos de confiança são criados para determinar quando uma instância é *aceitável* ou não. Intervalos são construídos em torno da precisão de classificação da instância atual (i.e., sua porcentagem de tentativas de classificação correta) e a freqüência observada de sua classe (i.e., a porcentagem de instâncias de treinamento processadas que são elementos dessa classe). O IB3 aceita uma instância se a sua precisão de classificação for significativamente maior que a sua freqüência observada de classe. O IB3 remove uma instância da descrição do conceito se a sua precisão de classificação for significativamente menor. Quando os dois intervalos se sobrepõem, então a instância permanece na expressão do conceito mas não é usada em tentativas de classificação. Estas três situações são ilustradas na Figura 4.1 e podem ser resumidas como [Aha 97]:

- quando intervalos se sobrepõem, a instância é retida, mas não é usada para predição
- se o intervalo de precisão da instância está “acima” de seu intervalo de freqüência da classe, então essa instância é *aceitável* e é usada para predições. A *aceitabilidade* é verificada sempre que uma predição é feita
- se o intervalo de precisão da instância está “abaixo” de seu intervalo de freqüência da classe, então a instância é descartada do armazenamento

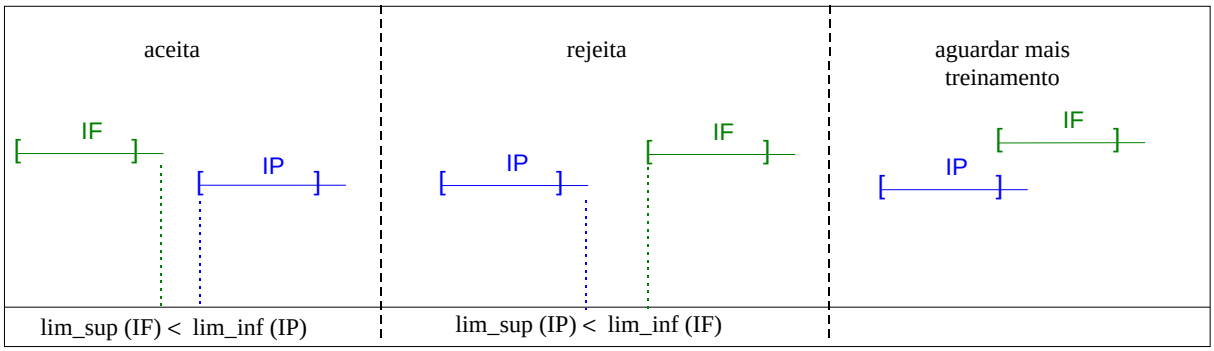

Figura 4.1. Situações de aceitação e descarte da instância, com base nos intervalos de confiança

O fato do IB3 marcar uma instância como *aceitável* não significa que tal instância permanecerá *aceitável* durante o resto do aprendizado. O IB3 repetidamente avalia a *aceitabilidade* de cada instância, com base em seus registros de classificação, à medida que o aprendizado prossegue.

O intervalo de precisão de classificação da instância (IP) é construído usando-se as fórmulas[4] :

- Limite Inferior(IP):

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

- Limite Superior(IP):

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

onde p é a precisão da instância observada e n é o número de tentativas de classificação da instância.

Para o intervalo de freqüência da classe observada (IF) são usadas:

- Limite Inferior(IF):

[4] Esta equação é encontrada em [Hogg 83], na pág. 286

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

- Limite Superior(IF):

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

onde p é a freqüência da classe observada e n é o número de instâncias de treinamento previamente processadas.

É importante lembrar que o IB3 mantém, para cada instância armazenada, um registro de classificação que traz o número de tentativas de classificação corretas e incorretas, ou seja, um registro do tipo <instância, #tentativas_class_correta, #tentativas_class_incorreta>. Usando esse registro, podem-se obter os valores de p e n para a construção do intervalo IP da instância. Já para a construção do IF, para cada classe, devem ser contabilizados a freqüência da classe (p) e o número de instâncias processadas previamente (n). O intervalo IF é o mesmo para instâncias da mesma classe. Durante o processo de treinamento, existe um intervalo de freqüência para cada uma das classes, i.e., se as instâncias de treinamento estão distribuídas entre 3 classes, existem apenas três diferentes intervalos de freqüência. Por outro lado, cada instância tem associada a ela um intervalo de precisão; existem pois tantos intervalos de precisão quantas forem as instâncias.

O teste com intervalos de freqüência e precisão foi projetado para tornar difícil a aceitação de uma instância. Para a aceitação da instância, z é tomado como 0.9 de maneira que sejam aceitas apenas aquelas instâncias que são “efetivamente” boas classificadoras. Por outro lado, z é tomado como 0.75 quando da eliminação da instância, com o objetivo de descartar até mesmo instâncias que tenham um desempenho moderado.

O IB3 tem uma justificativa para comparar a precisão das instâncias armazenadas com a freqüência relativa de suas classes. Este algoritmo normaliza a aceitação de uma instância com relação à distribuição da freqüência do conceito, com o objetivo de diminuir a sua sensibilidade a distribuições tendenciosas. Espera-se que instâncias de conceitos com alta freqüência relativa observada tenham também precisão de classificação relativamente alta, uma vez que uma porcentagem relativamente alta de suas tentativas de classificação serão de instâncias de suas classes, ou seja, serão bem sucedidas. De maneira análoga, espera-se que instâncias de conceitos com baixa freqüência relativa observada tenham precisão de classificação relativamente baixa. Comparando a precisão da instância com relação à freqüência de sua classe, o IB3 pode tolerar mais facilmente distribuições tendenciosas de instâncias do conceito. A atualização dos registros de classificação mantidos pelo IB3 é feita como mostrado na Tabela 4.1.

Para cada instância de treinamento t:

Encontrar seu vizinho *aceitável* mais próximo (de acordo com os intervalos de confiança construídos):

1. Se existe pelo menos uma instância *aceitável*, então os registros de classificação são atualizados para todas instâncias armazenadas que estão na região dada por uma hiperesfera cujo centro é a própria instância t e cujo raio é a distância normalizada entre t e o vizinho *aceitável* mais próximo de t. A Figura 4.2 mostra um exemplo, no espaço bidimensional composto de 11 instâncias (y_i, i=1,..,11), de quais instâncias terão seus registros de classificação atualizados, considerando-se uma instância de treinamento t. A Tabela 4.2 mostra as similaridades entre a nova instância t e as já existentes onde, nesse exemplo, as instâncias y_8, y_9 e y_{11} são instâncias *aceitáveis*. Como dentre as *aceitáveis* a instância mais próxima à instância t é a instância y_9 ($sim(t,y_9)=-1.3$), todas as instâncias com

   $$sim(t,y_i) \geq sim(t,y_9), \quad \text{onde } i=1,2,...,11$$

   terão seus registros de classificação atualizados. Assim sendo, as instâncias y_1, y_2, y_4, y_6 e y_9 terão seus registros atualizados.
2. Se nenhuma das instâncias armazenadas é *aceitável*, um número aleatório r pertencente ao intervalo [1,m] é gerado, onde m é o número de instâncias armazenadas. Assim, serão atualizados os registros de classificação das r instâncias mais similares a t. Essa situação é ilustrada na Figura 4.3 onde, no espaço bidimensional composto de 11 instâncias (y_i, i=1,..,11), não existe, nesse momento do aprendizado, nenhuma instância *aceitável*. Supondo que o número aleatório $r \in [1,11]$ gerado

seja 6, a 6ª instância com a maior similaridade será tratada como a instância *aceitável* mais similar a t. Na Tabela 4.3 pode ser visto que a 6ª instância mais similar a t é y_5. Assim, todas as instâncias y_i, i=1,..,11, tal que

$$sim(t,y_i) \geq sim(t,y_5)$$

terão seus registros de classificação atualizados, que nesse exemplo são as instâncias y_2, y_4, y_5, y_8, y_9, y_{10}.

Tabela 4.1. Atualização dos registros de classificação mantidos pelo IB3

É importante notar que no caso em que nenhuma das instâncias armazenadas é *aceitável*, o IB3, ao "eleger" uma delas *aceitável* através de uma escolha randômica, está simulando uma situação em que existe pelo menos uma instância *aceitável* e se comportando como tal. Ou seja, a existência ou não de instâncias *aceitáveis* não altera o seu comportamento.

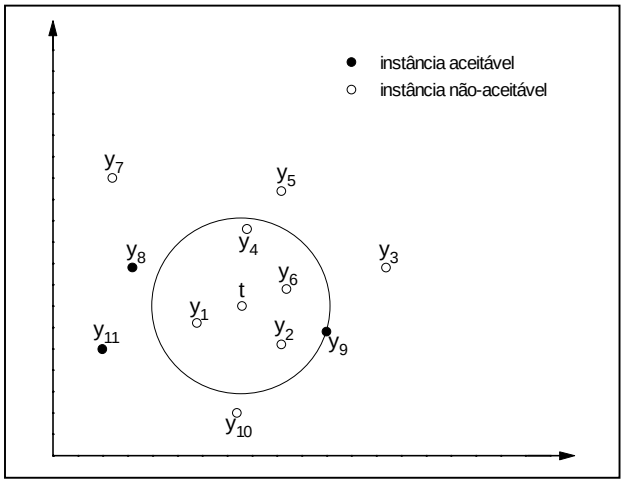

Figura 4.2. *Situação 1*: escolha, pelo IB3, das instâncias que terão seus registros de classificação atualizados

	Similaridades para a instância t	Similaridades em ordem crescente
$sim(t,y_1)$	-0.5	1ª
$sim(t,y_2)$	-0.8	3ª
$sim(t,y_3)$	-1.8	8ª
$sim(t,y_4)$	-1.0	4ª
$sim(t,y_5)$	-1.6	7ª
$sim(t,y_6)$	-0.7	2ª
$sim(t,y_7)$	-1.9	9ª
$sim(t,\mathbf{y_8})$	-1.6	7ª
$sim(t,\mathbf{y_9})$	**-1.3**	**5ª**
$sim(t,y_{10})$	-1.5	6ª
$sim(t,\mathbf{y_{11}})$	-1.8	8ª

Tabela 4.2. Tabela de similaridades das instâncias apresentadas na Figura 4.2

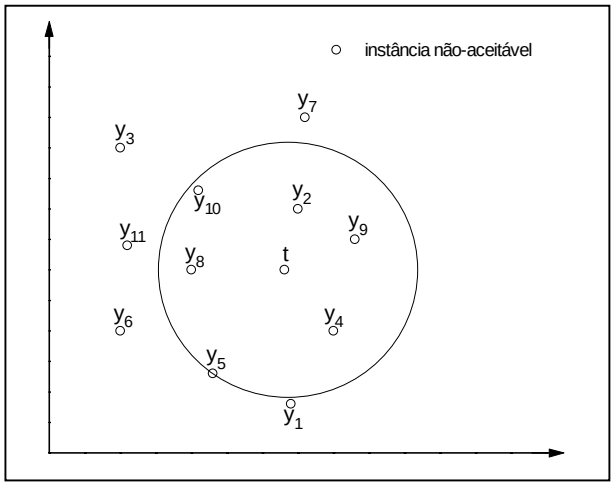

Figura 4.3. *Situação 2*: escolha, pelo IB3, das instâncias que terão seus registros de classificação atualizados

	Similaridades para a instância t	Similaridades em ordem crescente
$sim(t,y_1)$	-1.1	7ª
$sim(t,y_2)$	-0.55	1ª
$sim(t,y_3)$	-1.6	9ª
$sim(t,y_4)$	-0.7	4ª
$sim(t,\mathbf{y_5})$	**-0.9**	**6ª**
$sim(t,y_6)$	-1.3	8ª
$sim(t,y_7)$	-1.3	8ª
$sim(t,y_8)$	-0.6	2ª
$sim(t,y_9)$	-0.8	5ª
$sim(t,y_{10})$	-0.65	3ª
$sim(t,y_{11})$	-1.1	7ª

Tabela 4.3. Tabela de similaridades das instâncias apresentadas na Figura 4.3

Utilizando esses mecanismos de atualização dos registros de classificação e "filtragem" de instâncias com ruído, o algoritmo IB3 mostrou um melhor desempenho que os algoritmos IB1 e IB2, principalmente em domínios com grande quantidade de ruídos, pois a maioria das instâncias com ruído é descartada da descrição do conceito, não participando, então, da sua expressão final, a que é usada na fase de classificação. Segue-se a descrição do IB3, na Tabela 4.4.

DC ← ∅
for_each x∈ Conjunto de Treinamento ***do***
  begin
    1. ***for_each*** y∈ DC ***do***
        sim[y] ← similaridade(x, y)
    2. ***if*** ∃{y∈ DC | *aceitável*(y)}
        then y_{max} ← algum *aceitável* y∈ DC com a maior sim[y]
        else begin
                2.1. r ← valor aleatório ∈ [1, |DC|]
                2.2. y_{max} ← algum y∈ DC que é a r-ésima instância mais similar a y
            end
    3. ***if*** classe(x) = classe(y_{max})
        then classificação ← correta

```
        else begin
                3.1. classificação ← incorreta
                3.2. DC ← DC ∪ {x}
             end
    4. for_each y∈ DC do
        begin
            if sim[y] ≥ sim[y_max]
              then begin
                    4.1. Atualiza o registro de classificação de y
                    4.2. if registro de classificação de y é significativamente pobre
                          then DC ← DC - {y}
                   end
        end
end
```

Tabela 4.4. Algoritmo IB3 (DC = Descrição do Conceito)

Os percentuais de precisão e armazenamento obtidos pelos experimentos envolvendo o IB3, descritos em [Aha 91], são mostrados na Tabela 4.5 e comparados aos percentuais obtidos pelo algoritmo IB2. As características dos domínios de dados utilizados nos experimentos estão descritas nas Tabelas 3.4 e 3.6 mostradas previamente na Seção 3 (págs. 10 e 11, respectivamente). A análise dos resultados obtidos é mostrada na Tabela 4.6.

Domínios de dados	IB2		IB3	
Voting	90.9 ± 0.5	11.1	91.6 ± 0.5	7.4
Primary Tumor	32.9 ± 0.8	71.3	38.6 ± 0.9	16.4
LED Display	62.4 ± 0.6	41.5	71.7 ± 0.4	28.7
Waveform	69.6 ± 0.4	32.5	73.8 ± 0.4	14.6
Cleveland	71.4 ± 0.8	30.4	78.0 ± 0.8	7.7
Hungarian	55.9 ± 2.0	36.0	80.5 ± 0.9	7.5

Tabela 4.5. IB2 *versus* IB3: percentual de precisão ± desvio padrão e percentual de armazenamento

Domínios de dados	Análise dos resultados obtidos
Voting	• IB3 teve precisão de classificação similar aos algoritmos IB1 e IB2, apesar de exigir menos "armazenamento"
Primary Tumor	• IB3 teve uma precisão de classificação substancialmente superior à do IB2. A precisão inferior do IB2 se deve ao fato que, como o conjunto de dados é esparso, o algoritmo utilizou instâncias armazenadas de classes não muito freqüentes para classificar instâncias pertencentes a outras classes. Já o IB3, apesar de armazenar essas mesmas instâncias, só as utilizou em tentativas posteriores de classificação quando se mostraram serem boas classificadoras. Assim o IB3 raramente usa instâncias que descrevem classes com baixa freqüência para classificar incorretamente instâncias pertencentes a classes mais freqüentes
Led Display e Waveform	• nesses dois domínios de dados, o IB3 teve sua precisão significativamente aumentada quando comparada aos algoritmos IB1 e IB2. Além disso, a sua exigência de "armazenamento" foi também substancialmente menor. Esses resultados se devem ao fato que o IB3 filtra as instâncias que parecem ter ruído, descartando-as da descrição do conceito
Cleveland e Hungarian	• os melhores desempenhos do IB3 foram obtidos com estes dois domínios devido, principalmente, ao grande número de instâncias com ruído e instâncias "excepcionais" que são muito semelhantes a instâncias com ruído. A precisão de classificação do IB3 é bastante similar à do IB1, além de reduzir de forma expressiva as exigências de "armazenamento"

Tabela 4.6. Análise dos resultados obtidos pelo algoritmo IB3

Um outro domínio de dados utilizado nos experimentos evidenciou um problema relacionado ao IB3. No domínio *LED Display* com 24 atributos (dos quais 17 são irrelevantes), o IB3 mostrou-se bastante sensível a atributos irrelevantes. Os resultados da precisão de classificação dos algoritmos IB1, IB2 e IB3, para este domínio, são mostrados na Tabela 4.7. Nota-se, pelos valores da tabela, que os três algoritmos IB1, IB2 e IB3 não obtiveram uma boa precisão de classificação nesse domínio, devido à grande quantidade de atributos irrelevantes.

Domínios de dados	**IB1 (%)**		**IB2 (%)**		**IB3 (%)**	
LED Display (24 atributos)	47.2	100	42.0	60.5	45.8	25.8

Tabela 4.7. Percentual de precisão e percentual de armazenamento

Com o objetivo de contornar esse problema, visando promover um melhor desempenho do IB3, foi proposta uma versão que recebeu o nome de IB4 e que busca aprender sobre a relevância dos atributos.

É importante notar que as versões IB1, IB2 e IB3 assumem que todos os atributos que descrevem as instâncias de treinamento são igualmente relevantes na descrição do conceito. Isso, de certa maneira, faz com que o desempenho do IB3 degrade rapidamente à medida que atributos irrelevantes são usados na descrição das instâncias de treinamento; essa foi a razão que motivou o “aprendizado” da relevância de atributos da versão IB4.

5 Algoritmo IB4

O IB4 é 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.

Os experimentos descritos em [Aha 92] mostram que o IB4 é menos afetado pela presença de atributos irrelevantes do que é o IB3. No algoritmo IB4, para cada conceito, existe um conjunto de pesos dos atributos que são usados como parâmetros na função de similaridade. Todos os pesos dos atributos são atualizados após cada instância de treinamento ser classificada. O IB4 difere do IB3 com relação a :

- *função de similaridade* — é dependente do conceito. O cálculo da similaridade entre as instâncias x e y, para determinar a classificação de x com relação a um determinado conceito c, usa a definição mostrada na Tabela 5.1.

$$\text{similaridade}(c, x, y) = -\sqrt{\sum_{i=1}^{n} \text{peso}_{c_i}^2 * f(x_i, y_i)}$$

onde peso_{c_i} é o peso do atributo i, no conceito c

Tabela 5.1. Função de similaridade adotada pelo algoritmo IB4

  O fato da função de similaridade ser dependente do conceito pode ser ilustrado pelo exemplo descrito em [Aha 92]. No exemplo, para qualquer tigre t e gato g, a similaridade(*animal*,t,g) deve ser maior que similaridade(*animal_estimacao*,t,g). O IB4 aprende funções de similaridade "customizadas" para cada conceito
- *função de classificação* — no IB4 as instâncias são classificadas com relação a um determinado conceito. Assim, cada descrição de um conceito agrupa todas as instâncias incorretas juntas, independente de suas outras classificações com relação a outros conceitos
- *atualizador da descrição do conceito* — aprende pesos dos atributos para cada conceito através de um mecanismo onde, toda vez que uma instância x é classificada, o seu vizinho mais próximo y, com relação à descrição do conceito c, é usado para atualizar os pesos. A Tabela 5.2 descreve o algoritmo IB4, incluindo o esquema de atualização dos

pesos dos atributos usado pelo IB4 (Passo 5). Neste esquema, os atributos de maior relevância para um determinado conceito terão pesos maiores

DC ← ∅
for_each x∈ Conjunto de Treinamento ***do***
begin
1. ***for_each*** y∈ DC ***do***
sim[y] ← similaridade(x, y)
2. ***if*** ∃{y∈ DC | *aceitável*(y)}
then y_{max} ← algum *aceitável* y∈ DC com a maior sim[y]
else begin
2.1. r ← valor aleatório ∈ [1, |DC|]
2.2. y_{max} ← algum y∈ DC que é a r-ésima instância mais similar a y
end
3. ***if*** classe(x) = classe(y_{max})
then classificação ← correta
else begin
3.1. classificação ← incorreta
3.2. DC ← DC ∪ {x}
end
4. ***for_each*** y∈ DC ***do***
begin
if sim[y] ≥ sim[y_{max}]
then begin
4.1. Atualiza o registro de classificação de y
4.2. ***if*** registro de classificação de y é significativamente pobre
then DC ← DC - {y}
end
end
5. ***for_each*** atributo i ***do***
begin
5.1. diferença = | x_i - y_{max_i} |
5.2. ***if*** classe(x) = classe(y_{max})
then pesoacumulado $_{c_i}$ = pesoacumulado $_{c_i}$ + (1 - λ) ∗ (1 - diferença)
else pesoacumulado $_{c_i}$ = pesoacumulado $_{c_i}$ + (1 - λ) ∗ (diferença)
5.3. pesonormalizado $_{c_i}$ = pesonormalizado $_{c_i}$ + (1 - λ)
5.4. peso $_{c_i}$ = $\max\left(\frac{\text{pesoacumulado}_{c_i}}{\text{pesonormalizado}_{c_i}} - 0.5,\ 0\right)$
end
end

Tabela 5.2. Algoritmo IB4, onde x — instância sendo classificada; y_{max} — vizinho *aceitável* mais próximo; c — conceito alvo; λ — o maior valor entre a freqüência da classe de x e a freqüência da classe de y_{max}

Para mostrar o funcionamento do mecanismo de pesos usado pelo IB4, é apresentado um exemplo para aprender o conceito "voa" a partir de instâncias que são descritas por 3 atributos booleanos ("tem asas", "bípede", "tem penas"), cujos passos são mostrados nas Tabelas 5.3 e 5.4. Três instâncias são inicialmente apresentadas ao IB4, onde apenas uma tem a classe "voa", contendo os valores de atributos (verdadeiro, verdadeiro, verdadeiro). Suponha que os valores do *pesoacumulado* obtidos para cada atributo tenham sido (0.45, 0.45, 0.45) e o *pesonormalizado* (0.6). Se uma quarta instância, com valores de atributos (falso, verdadeiro, falso), é apresentada ao algoritmo e esta é incorretamente classificada como um animal que voa, então os novos *pesoacumulado* são (0.78, 0.45, 0.78) e o novo *pesonormalizado* é (0.93). Após o aprendizado da quarta instância os atributos recebem os pesos (0.34, 0, 0.34). Supondo, então, que uma quinta instância, contendo os valores de atributos (verdadeiro, verdadeiro, falso), seja corretamente classificada, são obtidos os valores (1.53, 1.2, 0.78) para os novos *pesoacumulado* e (1.68) para o *pesonormalizado*. Os pesos de cada atributo passam a ser (0.41, 0.22, 0). Isto significa que o IB4 aprendeu que o atributo "tem asas" é o mais relevante dos três atributos para o conceito "voa" e o atributo de menor relevância é o atributo "tem penas".

4ª instância: (falso, verdadeiro, falso)	
freqüência da classe de x : 2/3 freqüência da classe de y_{max} : 1/3 λ = max (2/3, 1/3) = 2/3 = 0.67	
pesonormalizado c_1 = pesonormalizado c_2 = pesonormalizado c_3 = 0.6 + (1-0.67) = 0.93	
Atributo 1:	diferença = | 0 - 1 | = 1 pesoacumulado c_1 = 0.45 + (1 - 0.67) ∗ 1 = 0.78 peso c_1 = $\max\left(\frac{0.78}{0.93} - 0.5,\ 0\right)$ = **0.34**
Atributo 2:	diferença = | 1 - 1 | = 0 pesoacumulado c_2 = 0.45 + (1 - 0.67) ∗ 0 = 0.45 peso c_2 = $\max\left(\frac{0.45}{0.93} - 0.5,\ 0\right)$ = **0**
Atributo 3:	diferença = | 0 - 1 | = 1 pesoacumulado c_3 = 0.45 + (1 - 0.67) ∗ 1 = 0.78 peso c_3 = $\max\left(\frac{0.78}{0.93} - 0.5,\ 0\right)$ = **0.34**

Tabela 5.3. Pesos dos atributos após a entrada da quarta instância de treinamento

5ª instância: (verdadeiro, verdadeiro, falso)	
freqüência da classe de x : 1/4 freqüência da classe de y_{max} : 1/4 λ = max (1/4, 1/4) = 1/4 = 0.25	
pesonormalizado c_1 = pesonormalizado c_2 = pesonormalizado c_3 = 0.93 + (1-0.25) = 1.68	
Atributo 1:	diferença = | 1 - 1 | = 0 pesoacumulado c_1 = 0.78 + (1 - 0.25) ∗ (1-0) = 1.53

	peso $_{c_1}$ = $\max\left(\frac{1.53}{1.68} - 0.5,\ 0\right)$ = **0.41**
Atributo 2:	diferença = | 1 - 1 | = 0 pesoacumulado $_{c_2}$ = 0.45 + (1 - 0.25) ∗ (1-0) = 1.2 peso $_{c_2}$ = $\max\left(\frac{1.2}{1.68} - 0.5,\ 0\right)$ = **0.22**
Atributo 3:	diferença = | 0 - 1 | = 1 pesoacumulado $_{c_3}$ = 0.78 + (1 - 0.25) ∗ (1-1) = 0.78 peso $_{c_3}$ = $\max\left(\frac{0.78}{1.68} - 0.5,\ 0\right)$ = **0**

Tabela 5.4. Pesos dos atributos após a entrada da quinta instância de treinamento

Como comentado anteriormente na Seção 3 (pág. 8), o IB4 utiliza um método para tratar instâncias com valores ausentes de atributos. Entretanto esse método traz um problema para o IB4, pois ao atualizar os pesos dos atributos com valores ausentes o IB4 “aprende” um conjunto de pesos de atributos, eventualmente incorretos, uma vez que o algoritmo assume valores para esses atributos que nem sempre refletem uma situação plausível. Assim que as instâncias com os novos atributos são introduzidas, o IB4 precisa novamente aprender os pesos dos atributos, agora de maneira correta, após estas instâncias serem processadas, o que exige um grande tempo de processamento do algoritmo. A proposta do IB5, discutida a seguir, é contornar esse problema, permitindo que novos atributos sejam tolerados de maneira mais eficiente.

6 Algoritmo IB5

Em situações de aprendizado comumente encontradas, nem sempre as instâncias podem ser inicialmente descritas por todos os possíveis atributos. Muitas vezes valores ausentes de atributos são introduzidos após o início do processo de aprendizado; é interessante que algoritmos de aprendizado permitam a introdução desses valores para que eles possam ser incorporados à expressão do conceito.

O IB5, extensão do IB4, mostra uma reação significativamente mais rápida à posterior introdução de atributos relevantes durante o processo de treinamento. Essa reação foi notificada aplicando os algoritmos IB4 e IB5 a um domínio de dados artificial contendo seis atributos com valores booleanos (dos quais apenas um atributo é relevante) e a classe associada. Estes experimentos, extraídos de [Aha 92], são comentados a seguir. A Figura 6.1 permite visualizar a rápida adaptação do IB5 à introdução de novos atributos relevantes. Nesta figura são descritas as curvas de aprendizado, relacionadas ao domínio artificial, para o IB4 e o IB5. As curvas mostram que a precisão de classificação dos dois algoritmos atingiu apenas 50% ao serem processadas as 100 primeiras instâncias. Isso ocorreu porque o único atributo relevante para predizer o conceito estava ausente nessas 100 instâncias.

Após processar mais 25 instâncias, estas agora contendo o valor do atributo relevante, o IB4 não atingiu 60% na precisão enquanto o IB5 aproximou-se dos 90%. Notou-se então, que a reação do IB5 foi mais rápida à introdução do novo atributo relevante. Essa reação mais rápida do IB5 é explicada considerando-se o mecanismo de atualização dos pesos dos atributos.

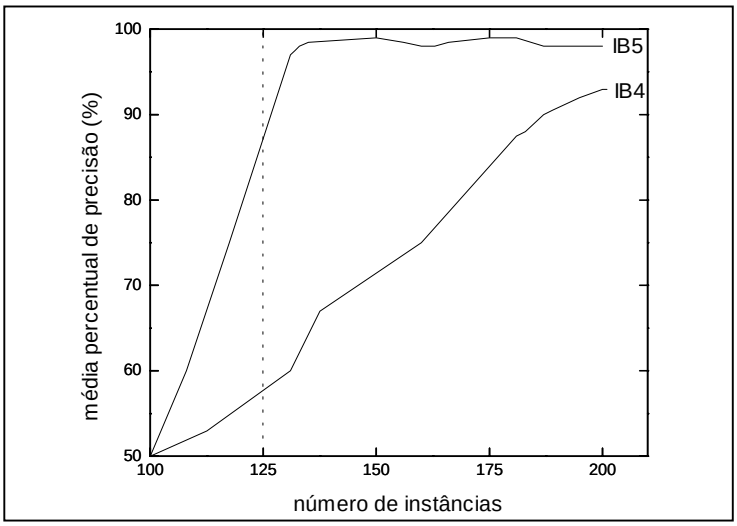

Figura 6.1. Adaptação mais rápida do IB5, com relação ao IB4, à introdução de novos atributos relevantes

O IB4 reagiu mais lentamente à introdução de novos atributos porque ele aprendeu um conjunto incorreto para o peso do atributo. Após as 100 primeiras instâncias, este conjunto incorreto de pesos fez com que o IB4 necessitasse de muito mais processamento, à medida que as próximas instâncias (que introduziram o novo atributo) foram processadas, pois aquele conjunto incorreto de peso dos atributos teve que ser reaprendido.

Já o IB5 teve uma rápida reação porque usa um método para tolerar novos atributos, onde apenas atributos cujos valores são conhecidos durante as tentativas de classificação são considerados, ignorando atributos cujos valores estão ausentes. Ao adotar tal procedimento o IB5 não precisa reaprender os pesos dos atributos.

O IB5 utiliza apenas valores conhecidos de atributos durante as tentativas de classificação. Ao calcular a similaridade entre as instâncias x e y, com relação ao conceito c, se o valor de um determinado atributo estiver ausente, seja em x ou em y, esse atributo é ignorado pela função, como mostra a definição na Tabela 6.1.

$$\text{similaridade}(c,x,y) = -\sqrt{\sum_{i=1}^{n} \text{peso}_{c_i}^2 * \text{dif}(x_i,y_i)^2}$$

onde

$$\text{dif}(x_i,y_i) = \begin{cases} |x_i - y_i| & \text{se ambosconhecidos}(x_i,y_i) = 1 \\ 0 & \text{caso contrário} \end{cases}$$

- ambosconhecidos(x_i,y_i) = 1 se ambos os valores de atributo, tanto em x quanto em y, são conhecidos, e 0 caso contrário

Tabela 6.1. Função de similaridade adotada pelo algoritmo IB5

O IB5 é praticamente idêntico ao algoritmo IB4 descrito na Seção 5 — Tabela 5.2 (pág. 22). As diferenças estão na função de similaridade e no Passo 5 do algoritmo. No IB5, os passos 5.1 a 5.4 só são processados se ambosconhecidos(x_i,y_i) = 1 e a função usada para o cálculo da similaridade é aquela definida na Tabela 6.1.

O fator determinante para a reação mais rápida do IB5 frente ao IB4 é que o IB5 apenas atualiza pesos dos atributos quando os valores dos atributos, tanto em x quanto em y, são conhecidos. Isto permite que os pesos dos atributos sejam aprendidos corretamente e que na presença de novos atributos não seja necessário "aprender" novamente os pesos dos atributos, como acontece no IB4. Desta forma, como os pesos dos atributos serão corretamente aprendidos, o uso desses pesos no cálculo da função de similaridade permitirá similaridades mais plausíveis e precisas.

7 Resumo Geral

As Tabelas 7.1 e 7.2 apresentam um resumo das principais características dos algoritmos da família IBL abordados nas seções anteriores.

8 Conclusões

Foram apresentados/discutidos, em detalhes, quatro algoritmos da família IBL, objetivando fornecer subsídios para uma caracterização precisa do paradigma de *aprendizado indutivo baseado em instâncias*.

É fato que a família IBL de aprendizado é limitada, pela própria natureza de seus algoritmos: armazenam pontos como expressão de conceito e não promovem qualquer generalização daquela expressão. Entretanto, como visto neste documento, tais algoritmos, em determinadas condições, podem ter um bom desempenho em determinados domínios.

É importante lembrar que problemas, como os evidenciados quando da apresentação das diversas versões do IBL, devem ser tratados para que sistemas de aprendizado adquiram maior robustez e precisão. Tratamentos de ruídos, da relevância incerta dos atributos, de atributos ausentes e da adaptação de novos atributos devem ser incorporados a um sistema de

aprendizado, devido ao fato que esses problemas, se não forem devidamente tratados, podem acarretar, entre outros, a redução no desempenho do sistema e o aumento na necessidade de espaço de armazenamento. As diversas variantes da família IBL permitiram mostrar que melhores desempenhos e menores quantidades de espaço de armazenamento podem ser conseguidos com a incorporação de algumas técnicas especializadas.

Referências

[Aha 91] Aha, D.W.; Kibler, D. & Albert, M.K. Instance-based learning algorithms. *Machine Learning 6(1)*, 1991, pp 37-66.

[Aha 92] Aha, D.W. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *Int. J. Man-Machine Studies 36*, 1992, pp 267-287.

[Aha 97] Comunicação Pessoal, 13/06/97.

[Beale 94] Beale, R. & Jackson, T. *Neural computing: an introduction*. Institute of Physics Publishing, 1994.

[Boulos 87] Boulos, P.; Camargo, I. *Geometria analítica: um tratamento vetorial*. Mc Graw-Hill, São Paulo, 1987.

[Breiman 84] Breiman, L.; Friedman, J.H.; Olshen, R.A. & Stone, C.J. Classification and regression trees. *Machine Learning 10*, 1984, pp 57-78.

[Cover 67] Cover, T. & Hart, P. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory 13*, 1967, pp 21-27.

[Gates 72] Gates, G.W. The reduced nearest neighbor rule. *IEEE Transactions on Information Theory 18*, 1972, pp 431-433.

[Hart 68] Hart, P.E. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory 14*, 1968, pp 515-516.

[Hogg 83] Hogg, R.V. & Tanis, E.A. *Probability and statistical inference*. New York, NY: Macmillan, 1983.

[Murphy 95] Murphy, P.M. & Aha, D.W.; *UCI Repository of machine learning databases.* (http://www.ics.uci.edu/AI/ML/MLDBRepository.html) University of California, Irvine, 1995.