Uma Avaliação de Dados de Expressão Gênica em Leucemias Agudas Utilizando Árvores de Decisão

Rogério Nunes Lemos[1,2] & José Augusto Baranauskas[1]

Universidade de São Paulo
[1]Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto
Departamento de Física e Matemática
[2]Faculdade de Medicina de Ribeirão Preto

Resumo: Um problema das análises de expressão gênica é que os resultados obtidos assumem proporções gigantescas, nos quais são encontrados milhares de genes diferencialmente expressos. Uma tentativa para se resolver este tipo problema pode ser o uso de sistemas que são capazes de adquirir conhecimento de forma automática a partir de grandes volumes de dados e contribuem com a geração de hipóteses úteis, por exemplo, na identificação de um subconjunto de genes que podem ser utilizados para fins de diagnóstico na prática clínica em casos de leucemia. Neste trabalho é avaliado um conjunto de dados de expressão gênica sobre dois tipos de leucemia usando árvores de decisão com arredondamento de valores.

Palavras-chave: Aprendizado de Máquina, Árvore de Decisão, Expressão Gênica, Leucemia.

Abstract: An important issue about gene expression analysis concerns the huge amount of generated results: normally thousands of differently expressed genes can be found. One possible way to solve this problem consists in applying systems that are able to extract knowledge automatically from large datasets yielding to useful hypotheses. For instance, a gene subset could be identified in order to help diagnosing leukemia cases at clinical practice. This work evaluates data from leukemia gene expression through decision trees using rounding of continuous values.

Key-words: Machine Learning, Decision Tree, Gene Expression, Leukemia.

Introdução

Nas últimas décadas, a computação científica e comercial vem gerando uma quantidade enorme de dados. Métodos tradicionais de manipulação de dados, tais como planilhas, consultas em bancos de dados, programas gráficos e processadores de texto são ferramentas úteis para o armazenamento, gerenciamento e a organização de dados e informações. Entretanto, quando se trata de descoberta do conhecimento existente, por exemplo, em um banco de dados, torna-se necessário recorrer a outras estratégias.

A extração semi-automática de conhecimento a partir de grandes volumes (bancos) de dados — KDD (*Knowledge Data Discovery*) — é um ramo de pesquisa que tem como principais objetivos a aplicação e o desenvolvimento de técnicas e ferramentas que automatizem o processo de manipulação de dados, visando a extração de novas informações úteis. Uma das abordagens utilizada consiste em utilizar algoritmos de Aprendizado de Máquina — AM.

O Aprendizado de Máquina supervisionado é definido por [1] como "Um sistema de aprendizado é um programa de computador que toma decisões baseadas na experiência contida em exemplos solucionados com sucesso."

No Aprendizado de Máquina supervisionado, cada exemplo z pode ser descrito por um vetor de valores de características x, ou atributos, juntamente com o rótulo da classe associada y ou seja, $z = (x, y)$, ficando subentendido o fato que tanto x como z são vetores, ou seja, $\overrightarrow{z} = (\overrightarrow{x}, y)$. Para rótulos de classe y discretos, esse problema é conhecido como *classificação* e para valores contínuos como *regressão*.

O objetivo de um algoritmo de AM, denominado *indutor*, é construir uma hipótese $h(\cdot)$ que possa determinar corretamente a classe de novos exemplos ainda não rotulados, ou seja, exemplos que não tenham o rótulo da classe. Formalmente, em classificação, um exemplo z é um par $(x, y) = (x, f(x))$ onde x é a entrada e $f(x)$ é a saída e $y = f(x)$. A tarefa de um indutor é, dado um conjunto de exemplos da função $f(\cdot)$, induzir uma função $h(\cdot)$ que aproxima $f(\cdot)$, normalmente desconhecida. Neste caso, $h(\cdot)$ é chamada uma *hipótese* sobre a função objetivo $f(\cdot)$, ou seja, $h(x) \approx f(x)$.

Dentre os algoritmos de AM supervisionado uti-

lizando classificação existe uma família de algoritmos de AM indutivo conhecida como *Top Down Induction of Decision Trees* — TDIDT. De modo simplificado a indução de uma árvore de decisão realiza-se da seguinte forma [2, 3]: utilizando o conjunto de treinamento, um atributo é escolhido de forma a particionar os exemplos em subconjuntos, de acordo com valores deste atributo. Para cada subconjunto, outro atributo é escolhido para particionar novamente cada um deles. Este processo prossegue, enquanto um dos subconjuntos contenha uma mistura de exemplos pertencendo a classes diferentes. Uma vez obtido um subconjunto uniforme — todos os exemplos naquele subconjunto pertencem à mesma classe — um nó folha é criado e rotulado com o mesmo nome da respectiva classe.

Quando um novo exemplo deve ser classificado, começando pela raiz da árvore induzida, o classificador testa e desvia para cada nó com o respectivo atributo até que atinja uma folha. A classe deste nó folha será então atribuída ao novo exemplo.

Outro fator importante que também deve ser considerado é o grau de compreensibilidade proporcionado ao ser humano. De acordo com [4] e [5], os sistemas de aprendizado são classificados em duas grandes categorias:

1. sistemas *caixa-preta* que desenvolvem sua própria representação do conceito, isto é, sua representação interna pode não ser facilmente interpretada por humanos e não fornecem nem esclarecimento, nem explicação do processo de reconhecimento;

2. sistemas *orientados a conhecimento* que objetivam a criação de estruturas simbólicas que sejam compreensíveis por humanos.

Assim, no aprendizado de conceitos, o interesse principal consiste em obter descrições simbólicas que sejam fáceis de serem compreendidas e utilizadas por meio de modelos mentais. Segundo o *postulado da compreensibilidade* de [6]: "Os resultados da indução por computador devem ser descrições simbólicas das entidades dadas, sendo semântica e estruturalmente similares àquelas que um especialista humano poderia produzir observando as mesmas entidades. Os componentes dessas descrições devem ser compreensíveis como simples 'pedaços' de informação, diretamente interpretáveis em linguagem natural, bem como reportar conceitos quantitativos e qualitativos de maneira integrada."

Como regra prática, [6] assume que os componentes de descrição, tais como regras ou nós em uma árvore de decisão, devem ser expressões contendo menos de cinco condições em uma conjunção; poucas condições em uma disjunção; no máximo um nível de parênteses; no máximo uma implicação; não mais de dois quantificadores e nenhuma recursão. Embora esses valores possam ser flexíveis, descrições geradas por indução dentro dos limites propostos são similares à representação do conhecimento humano e, portanto, fáceis de serem compreendidas. Embora tais medidas sejam simples de serem avaliadas, é importante salientar que elas são meramente sintáticas e que, muitas vezes, medidas semânticas devam ser consideradas [7].

Em Aprendizado de Máquina existem muitos algoritmos de aprendizado que induzem classificadores. Este trabalho se concentra em indutores que contribuem para a compreensão dos dados em contraste com indutores que visam apenas uma grande precisão. Por exemplo, a indução de regras ou árvores de decisão pode auxiliar médicos a compreenderem melhor os dados, enquanto uma rede neural convencional, mesmo com precisão similar, pode ser extremamente difícil de ser compreendida por seres humanos[1]. Por exemplo, no desenvolvimento de sistemas especialistas é importante que especialistas humanos possam, fácil e confiavelmente, verificar o conhecimento extraído e relacioná-lo ao seu próprio domínio de conhecimento. Além disso, algoritmos de aprendizado que induzem estruturas compreensíveis, contribuindo para a compreensão do domínio considerado, podem produzir conhecimento novo [8].

Um ponto importante é que enquanto a maior parte das operações para construir uma árvore de decisão cresce linearmente com o número de exemplos de treinamento, o processo de escolha de um atributo contínuo contendo d valores distintos requer a ordenação desses valores, crescendo como $d \log_2 d$ [9]. Assim, o tempo requerido para construir uma árvore de decisão a partir de um conjunto de treinamento grande pode ser dominado pela ordenação de atributos contínuos, por exemplo, os algoritmos C4.5 [9] e J48 [10] fazem uso do algoritmo *quicksort* para ordenar valores contínuos [11][Cap. 7], [12][Cap. 2].

O objetivo deste trabalho consiste na avaliação do arredondamento de valores de expressão gênica em leucemias agudas usando árvores de decisão, ou seja, neste trabalho é tratado o aprendizado simbólico supervisionado para resolver problemas de classificação. O termo *simbólico* indica que os classificadores devem ser legíveis e interpretáveis por humanos. O termo *supervisionado* sugere que algum processo, às vezes denominado *agente externo* ou *professor*, previamente rotulou os dados. Finalmente, o termo *classificação* denota o fato que o rótulo da classe é discreto, ou seja, consiste de valores nominais sem uma ordem definida. Nesta pesquisa é utilizado o indutor de árvores de decisão J48 da biblioteca Weka [10] − Waikato Environment for Kno-

[1] Existem, entretanto, vários métodos desenvolvidos para a extração de regras a partir de redes neurais.

wledge Analysis, uma reimplementação na linguagem Java do indutor C4.5 [9].

Conjunto de Exemplos

O conjunto de exemplos utilizado (*aml-all*) corresponde a dados de expressão gênica obtidos por monitoramento de *microarrays* de DNA. O problema consiste em distinguir entre a leucemia linfoblástica aguda (*acute lymphoblastic leukemia* - ALL) e leucemia mielóide aguda (*acute myeloid leukemia* - AML). No trabalho desenvolvido por [13] o conjunto de treinamento possui 38 exemplos (27 do tipo ALL e 11 do tipo AML) e o conjunto de teste possui 34 exemplos (20 do tipo ALL e 14 do tipo AML). Todos exemplos são descritos por valores de expressão de 7129 genes. Adicionalmente, outro artigo que utiliza esse conjunto de exemplos é [14].

Metodologia

Neste trabalho foi avaliada a técnica de arredondamento proposta por [15], descrita em maiores detalhes a seguir. Inicialmente, considere uma variável ix inteira a ser arredondada e o fragmento de código expresso na Equação (1) onde k é o número de casas decimais mais à direita do número a ser arredondado. A função $int(x)$ retorna a parte inteira de x — por exemplo, $int(3.0) = 3$; $int(3.5) = 3$; $int(3.8) = 3$ — e a função $mod(x, y)$ corresponde ao resto da divisão inteira de x por y — por exemplo, $mod(10, 3) = 1$; $mod(10, 4) = 2$; $mod(12, 5) = 2$. Assume-se que a divisão retorna sempre um valor real, mesmo que seus argumentos sejam inteiros — por exemplo $2/4 = 0,5$; $1/4 = 0,25$. A variável iy é inteira.

$$\begin{aligned} &iy \leftarrow int(ix/10^k) \\ &\textbf{if}(mod(ix, 10^k) \geq 10^k/2) \textbf{ then } iy \leftarrow iy + 1 \quad \textbf{endif} \\ &ix \leftarrow iy \times 10^k \end{aligned} \tag{1}$$

A Equação (1) pode ser generalizada para qualquer base b além da base decimal, representada por meio da Equação (2).

$$\begin{aligned} &iy \leftarrow int(ix/b^k) \\ &\textbf{if}(mod(ix, b^k) \geq b^k/2) \textbf{ then } iy \leftarrow iy + 1 \quad \textbf{endif} \\ &ix \leftarrow iy \times b^k \end{aligned} \tag{2}$$

Em termos computacionais há interesse em utilizar base binária, ou seja, $b = 2$ por motivos de eficiência. Na base binária as divisões por 2 (ou potências de 2) podem ser efetuadas por meio de deslocamento (*shift*) de *bits* à direita e multiplicações por meio de deslocamento de *bits* à esquerda.

O tempo de arredondamento de um grande conjunto de dados é relativamente pequeno, segundo o Algoritmo 1 proposto por [15] que descreve o procedimento geral para arredondamento de valores de um atributo, no qual a Equação (2) corresponde às linhas 13–17. Admitindo um número máximo de valores max para cada atributo, os valores do atributo são ordenados, para que o número de valores distintos possam ser contados. A ordem é guardada e não são necessárias ordenações adicionais. Começando com $k = 1$, o valor de k é incrementado até o número de valores ser reduzido a um valor menor ou igual ao máximo desejado, max. Para que o Algoritmo 1 possa ser aplicado a um conjunto de exemplos, o processo deve ser repetido para cada atributo, como pode ser visto no Algoritmo 2.

Algoritmo 1 Algoritmo de arredondamento proposto por Weiss

Require: $\{v_i\}$, conjunto dos valores de um atributo
max, o máximo de valores distintos desejados
b, base a ser utilizada
Ensure: $\{v_i\}$ contendo no máximo max valores distintos
1: $s \leftarrow 1$
2: Se o conjunto $\{v_i\}$ contém frações, multiplica-se todos os valores por uma constante para que se obtenha apenas valores inteiros
3: Ordene os valores $\{v_i\}$
4: **loop**
5: $num \leftarrow$ número de valores distintos de $\{v_i\}$
6: **if** $num \leq max$ **then**
7: **exit loop**
8: **end if**
9: $s \leftarrow s + 1$
10: **for all** valores $ix \in \{v_i\}$ **do**
11: Se ix negativo, multiplicar por -1
12: $k \leftarrow s$
13: $iy \leftarrow$ int(ix/b^k)
14: **if** (mod$(ix, b^k) \geq b^k/2$) **then**
15: $iy \leftarrow iy + 1$
16: **end if**
17: $ix \leftarrow iy \times b^k$
18: Voltar o número ix para negativo se necessário
19: **end for**
20: **end loop**
21: Dividir todos os valores pela mesma constante utilizada no início para voltar as frações
22: **return** conjunto arredondado $\{v_i\}$

Os Algoritmos 1 e 2 foram implementados na linguagem de programação Java [16] para a realização de experimentos descritos nesta Seção. Note, entretanto, que as linhas 3 e 5 do Algoritmo 1 são desnecessárias,

Algoritmo 2 Algoritmo final de arredondamento

Require: $dataset$, conjunto de exemplos
p, porcentagem máxima de valores distintos
b, base a ser utilizada

1: **for all** atributo $v_i \in dataset$ **do**
2: Ordene os valores $\{v_i\}$
3: $num \leftarrow$ número de valores distintos de $\{v_i\}$
4: $max \leftarrow num \times p$
5: **Execute** $Algoritmo\ \ 1$ com parâmetros $\{v_i\}, max, b$
6: **end for**
7: **return** conjunto de exemplos arredondado

caso o mesmo seja executado pelo Algoritmo 2.

No experimento realizado foram avaliados tempo de indução, taxa de erro, e tamanho do classificador usando *holdout*[2]. Esta metodologia foi utilizada nesse conjunto por [13] e, posteriormente, também utilizada por [14].

Para os exemplos *aml-all*, no conjunto de treinamento contendo 38 exemplos foi aplicado arredondamento dos valores, deixando intacto o conjunto de teste que contém 34 exemplos.

Nos próximos parágrafos é freqüentemente mencionado o Algoritmo 1 por se tratar do algoritmo originalmente proposto por [15], embora, em termos computacionais, o Algoritmo 2 tenha sido, de fato, utilizado.

Como já mencionado, o Algoritmo 1 possui o parâmetro (p) que indica a porcentagem máxima permitida de valores distintos que são obtidos após aplicação do arredondamento no conjunto original, para cada atributo. Por exemplo, para um conjunto com 2 atributos, sendo o primeiro atributo contendo 100 valores distintos e o segundo atributo contendo 200 valores distintos, após a execução do Algoritmo 1 o conjunto derivado para $p = 50\%$ terá, no máximo, 50 valores distintos para o primeiro atributo e 100 valores distintos para o segundo atributo.

Neste experimento foram utilizados os valores de p iguais a 90%, 80%, 70%, 60% e 50%, obtendo um conjunto derivado de *aml-all* para cada valor de p.

Denomina-se conjunto original aquele que não se aplicou nenhum arredondamento (*aml-all*), e conjuntos derivados aqueles que sofreram arredondamento para 90%, 80%, 70%, 60% e 50% do total de valores, (*aml-all-90%*), (*aml-all-80%*), ..., (*aml-all-50%*), respectivamente.

Resultados e Discussão

Para avaliar o tempo de indução foram unificados os 38 exemplos de treinamento e 34 exemplos de teste e foi utilizado *10-fold stratified cross-validation* sobre os 72 exemplos. Para permitir comparações de nossos resultados com aqueles publicados na literatura, o classificador foi induzido apenas sobre o conjunto de 38 exemplos de treinamento, enquanto a taxa de erro foi avaliada utilizando-se o conjunto independente de 34 exemplos de teste.

Nas árvores induzidas a partir do conjunto de treinamento *aml-all*, aparece um único atributo, o gene *Zyxin*, tanto na árvore induzida a partir do conjunto original como aquelas induzidas a partir dos conjuntos derivados, como pode ser observado na Tabela 1. O tamanho da classificador é sempre constante para o conjunto original *aml-all* e todos seus derivados, e é igual a três.

Classificador	Árvore
aml-all	Zyxin $<=$ 938: ALL (27.0) Zyxin $>$ 938: AML (11.0)
aml-all-90%, base 2 *aml-all*-80%, base 2	Zyxin $<=$ 960: ALL (27.0) Zyxin $>$ 960: AML (11.0)
aml-all-70%, base 2 *aml-all*-60%, base 2 *aml-all*-50%, base 2	Zyxin $<=$ 1024: ALL (27.0) Zyxin $>$ 1024: AML (11.0)
aml-all-90%, base 10 *aml-all*-80%, base 10 *aml-all*-70%, base 10	Zyxin $<=$ 900: ALL (27.0) Zyxin $>$ 900: AML (11.0)
aml-all-60%, base 10 *aml-all*-50%, base 10	Zyxin $<=$ 1000: ALL (28.0/1.0) Zyxin $>$ 1000: AML (10.0)

Tabela 1: Classificador para o conjunto *aml-all* e derivados

A taxa de erro no conjunto de treinamento é sempre igual a zero para todos os conjuntos de exemplos, exceto para *aml-all-60%* e *aml-all-50%* utilizando base 10, cuja taxa de erro é de $1/38 = 2,63\%$. A taxa de erro no conjunto independente de teste é sempre igual a $3/34 = 8,82\%$ para todos os conjuntos de exemplos.

Na Tabela 2 são mostrados os resultados (média $\pm$ desvio padrão) obtidos em relação aos conjuntos de exemplos *aml-all* original e derivados. A segunda e terceira colunas representam os resultados do tempo de indução, utilizando a base binária e a base decimal, respectivamente.

Na Figura 1 é mostrada a diferença absoluta em desvios padrões do tempo de indução entre o conjunto original e os conjuntos derivados, ou seja, entre *aml-all* e *aml-all-90%*, entre *aml-all* e *aml-all-80%* e assim por

[2]Para uma revisão sobre métodos de amostragem e de avaliação de algoritmos vide [17][Cap. 4].

Conjunto	Tempo (s) (base 2)	Tempo (s) (base 10)
aml-all	$3,00 \pm 0,55$	$3,00 \pm 0,55$
aml-all-90%	$2,69 \pm 0,40$	$2,19 \pm 0,30$
aml-all-80%	$2,40 \pm 0,31$	$2,09 \pm 0,26$
aml-all-70%	$2,33 \pm 0,45$	$1,95 \pm 0,32$
aml-all-60%	$2,15 \pm 0,38$	$1,97 \pm 0,39$
aml-all-50%	$1,94 \pm 0,38$	$1,69 \pm 0,17$

Tabela 2: Tempo de indução do classificador utilizando arredondamento com bases 2 e 10 *aml-all*

diante, utilizando base 2. Quando a barra encontra-se acima de zero significa que o respectivo classificador do conjunto derivado supera o desempenho do classificador do conjunto original; se a barra encontra-se abaixo de zero então o classificador do conjunto original supera o respectivo classificador do conjunto derivado. Quando a altura da barra estiver acima (abaixo) de dois (menos dois) significa que o classificador do conjunto derivado (conjunto original) supera o classificador do conjunto original (conjunto derivado) significativamente, ou seja, nível de confiança de 95% [17, 18].

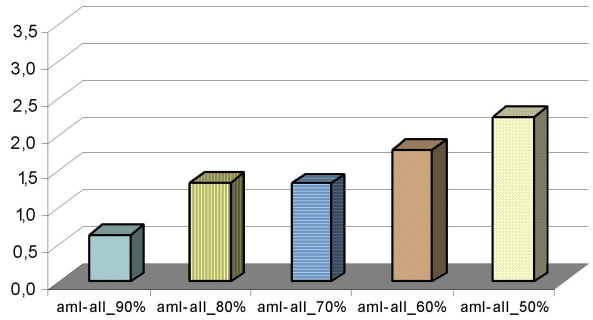

Figura 1: Diferença absoluta do tempo de indução (arredondamento utilizando base 2 *versus* conjunto original) *aml-all*

Nota-se que o tempo de indução reduziu para todos os conjuntos utilizando arredondamento, sendo de forma não significativa para todos os conjuntos, exceto *aml-all-50%* (com grau de confiança de 95%).

Na Figura 2 é mostrada a diferença absoluta em desvios padrões do tempo de indução entre o conjunto original e os conjuntos derivados, ou seja, entre *aml-all* e *aml-all-90%*, entre *aml-all* e *aml-all-80%* e assim por diante, utilizando base 10.

O tempo de indução reduziu para todos os conjuntos utilizando arredondamento, sendo de forma não significativa apenas para o conjunto *aml-all-90%*; todos os outros tiveram uma redução significativa.

A seguir é efetuada uma comparação entre os resultados obtidos neste trabalho com aqueles obtidos

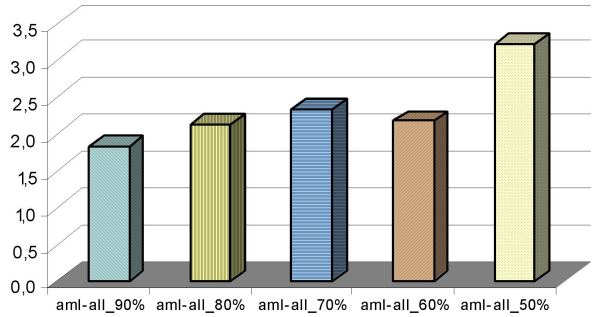

Figura 2: Diferença absoluta do tempo de indução (arredondamento utilizando base 10 *versus* conjunto original) *aml-all*

por [13] e [14], que abordam estratégias de voto ponderado e indução de regras, respectivamente. Nesta comparação, apenas os resultados obtidos a partir do conjunto independente de teste são considerados.

O classificador obtido por [13] apresenta taxa de erro de $5/34 = 14,70\%$ no conjunto de teste.

O classificador obtido por [14] que consiste de duas regras, ambas com duas condições cada, é similar em tamanho a uma árvore de decisão contendo 6 nós. Cada regra é considerada como um classificador separado pelos autores e, sendo assim, são reportadas duas taxas de erro no conjunto de teste de $7/34 = 20,59\%$ para a regra cuja conclusão é a classe AML e $2/34 = 5,88\%$ para a regra cuja conclusão é a classe ALL, ambas taxas calculadas sobre o conjunto de teste.

Comparados com os resultados relatados por [13] e [14], a taxa de erro obtida em nossos resultados é de $3/34 = 8,82\%$ um pouco menor do que aquelas obtidas por abordagem de voto ponderado. A árvore de decisão obtida também é ligeiramente menor do que as regras induzidas podendo ser mais facilmente interpretada.

Outro ponto interessante é que a árvore de decisão identificou como importante para a separação das classes AML e ALL um atributo (*Zyxin*) que também é reportado por [13] como sendo um "gene informativo", dentre outros.

Conclusões

É interessante notar que, embora abordagens mais sofisticadas tenham sido descritas na literatura para o conjunto de exemplos *aml-all*, a utilização de árvores de decisão parece ser promissora para a análise de dados de expressão gênica. Considerando a utilização de árvores de decisão com arredondamento também é possível notar que não houve alteração do

atributo selecionado, assim com o valor escolhido para o teste do atributo teve uma pequena oscilação, mesmo para 50% de arredondamento de valores. É possível que esta técnica possa ser útil para melhor definir os valores de teste escolhidos pela árvore de decisão dos níveis de expressão gênica.

Agradecimentos

À FAPESP, Fundação de Amparo à Pesquisa do Estado de São Paulo, pelo apoio financeiro (processo n^o 04/10277-0).

Referências

[1] S. M. Weiss and C. A. Kulikowski. *Computer Systems that Learn*. Morgan Kaufmann, San Mateo, CA, 1991.

[2] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth & Books, Pacific Grove, CA, 1984.

[3] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986. Reprinted in Shavlik and Dietterich (eds.), 1990. Readings in Machine Learning, Morgan Kaufmann Publishers, Inc.

[4] Ryszard S. Michalski. A theory and methodology of inductive learning. *Artificial Intelligence*, 20:111–161, 1983.

[5] Miroslav Kubat, Ivan Bratko, and Ryszard S Michalski. *A Review of Machine Learning Methods*, pages 3–69. John Wiley & Sons Ltd., West Sussex, England, 1998.

[6] R. S. Michalski. *A Theory and Methodology of Inductive Learning*, pages 83–134. Morgan Kaufmann, Los Altos, CA, 1983.

[7] M. J. Pazzani. Knowledge discovery from data? *IEEE Intelligent Systems*, 13:10–12, 2000. March/April 2000.

[8] T. G. Dietterich. Learning at the knowledge level. *Machine Learning*, 1(3):287–315, 1986. Reprinted in Shavlik and Dietterich (eds.), 1990. Readings in Machine Learning, Morgan Kaufmann Publishers, Inc.

[9] J. R. Quinlan. C4.5*: Programs for Machine Learning*. Morgan Kaufmann, 1993. San Francisco, CA.

[10] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, volume 1. Morgan Kaufmann, october 1999.

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Algoritmos: Teoria e Prática*. Campus, 2002. 2ª edição.

[12] Niklaus Wirth. *Algoritmos e Estruturas de Dados*. Prentice Hall do Brasil, 1986.

[13] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, , and E. S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

[14] Dragan Gamberger, Nada Lavrac, Filip Zelezny, and Jakub Tolar. Induction of comprehensible models for gene expression datasets by subgroup discovery methodology. *Journal of Biomedical Informatics*, 37:269–284, 2004.

[15] S. M. Weiss and N. Indurkhya. *Predictive Data Mining: A Practical Guide*. Morgan Kaufmann, San Francisco, CA, 1998.

[16] H M Deitel and P J Deitel, editors. *Java: Como Programar*. Prentice-Hall, 2005.

[17] Solange O Rezende, editor. *Sistemas Inteligentes: Fundamentos e Aplicações*. Manole, 2003.

[18] Lincoln E. Moses, editor. *Think and Explain with Statistics*. Addison–Wesley, 1986.

Contato

Universidade de São Paulo
Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto
Departamento de Física e Matemática
Avenida do Café, 3900
14040-901 — Ribeirão Preto, SP — Brasil
(16) 3602-3693
rnlemos@fmrp.usp.br
augusto@ffclrp.usp.br