Uma Proposta para Evoluir Classificadores Simbólicos Utilizando um Algoritmo Genético

Flávia Cristina Bernardini[*], **Maria Carolina Monard**
Laboratório de Inteligência Computacional
Instituto de Ciências Matemáticas e Computação
Universidade de São Paulo
Av. do Trabalhador Sancarlense, 400 – Caixa Postal 668
CEP 13560-970 São Carlos, SP
{fbernard,mcmonard}@icmc.usp.br

Abstract

Data Mining applications generally use learning algorithms in order to induce knowledge. In domains where explanation about classification decisions is essential, symbolic supervised learning algorithms are appropriated. To scale up learning algorithms to deal with large databases, data sampling techniques can be applied. Afterwards, learning algorithms can be used on each sample to induce a set of classifiers which can be combined into an ensemble of classifiers or into a unique classifier. In this work we consider the latter approach and propose the use of a genetic algorithm. We have implemented the genetic algorithm and several evaluation functions into a computational environment for evolving sets of knowledge rules, described in this work as well as experiments carried out on several datasets. Good experimental results were obtained by the genetic algorithm.

Keywords: Symbolic Machine Learning, Genetic Algorithm, Evolution Computation.

Resumo

Em aplicações práticas de Mineração de Dados geralmente são utilizados algoritmos de aprendizado para induzir conhecimento. Quando é necessário explicar as decisões de classificação dos classificadores induzidos, são indicados algoritmos de aprendizado simbólicos. Entretanto, a maioria dos algoritmos de aprendizado simbólico disponíveis não conseguem manipular grandes bases de dados. Uma maneira de tentar solucionar esse problema é induzir vários classificadores, utilizando diversas amostras da base de dados, e combiná-los em um *ensemble* de classificadores ou em um único classificador simbólico. Neste trabalho exploramos essa segunda solução, propondo o uso de algoritmos genéticos utilizando diversas funções de avaliação. Também, descrevemos o sistema computacional que implementa esse algoritmo e experimentos realizados com algumas bases de dados. Os resultados experimentais mostram que é possível evoluir bons classificadores utilizando o algoritmo genético proposto.

Palavras chave: Aprendizado de Máquina Simbólico, Algoritmo Genético, Computação Evolutiva.

1 INTRODUÇÃO

Em Aprendizado de Máquina — AM — supervisionado, é fornecido ao algoritmo de aprendizado um conjunto de exemplos rotulados para o algoritmo induzir um classificador, também denominado hipótese ou modelo, cujo objetivo é rotular novos exemplos. Assim, é importante avaliar a precisão do classificador, mas também é importante analisar o conhecimento nele contido, com o objetivo de descobrir novo conhecimento. Nesse caso, é fortemente indicado o uso de algoritmos de aprendizado simbólico, nos quais o modelo induzido pode ser diretamente interpretado pelo usuário/especialista do domínio. Porém, quando são utilizados grandes conjuntos de exemplos, *i.e.*, grandes bases de dados, como as utilizadas em mineração de dados, a maioria

[*]Trabalho realizado com auxílio da FAPESP, Brasil, Proc. Nº 02/06914-5.

dos algoritmos de aprendizado simbólico disponíveis não conseguem manipular tais conjuntos. Diversas soluções têm sido propostas para lidar com esse problema. Uma delas é utilizar a abordagem de *ensembles* de classificadores. Nessa abordagem, podem ser utilizados algoritmos de aprendizado já estabelecidos para induzir um conjunto de classificadores utilizando diversas amostras da base de dados. Após, é construído um *ensemble* que consiste do conjunto desses classificadores e cujas decisões são combinadas de alguma maneira para classificar novos exemplos. Uma outra solução é utilizar Algoritmos Genéticos — AGs —, por exemplo, para evoluir esse conjunto de classificadores em um único classificador final. Uma das vantagens de utilizar AGs está relacionada com o armazenamento do modelo (classificador) induzido para classificar novos exemplos, já que quando se utilizam *ensembles* de classificadores, todos os classificadores que participam do *ensemble* devem ser armazenados para realizar a classificação de novos exemplos, enquanto que usando AGs somente o classificador final deve ser armazenado.

Como mencionado, um outro aspecto importante relacionado ao uso de algoritmos de aprendizado no processo de mineração de dados tem a ver com a construção de modelos simbólicos, tal que sejam capazes de justificar a classificação de novos exemplos. A facilidade de explicação de um classificador é fundamental em várias áreas de conhecimento, tais como a área médica. Nesse caso, é apropriado utilizar algoritmos de aprendizado simbólico, os quais induzem classificadores que consistem de um conjunto de regras de conhecimento, facilmente interpretáveis pelos usuários, que permitem explicar a classificação de novos exemplos.

Neste trabalho, propomos o uso de AGs para evoluir um conjunto de classificadores simbólicos em um único classificador simbólico. Na nossa proposta, cada conjunto de regras que constituem um classificador, os quais formam indivíduos da população inicial do AG, pode ser induzido utilizando diversas amostras da base de dados bem como o mesmo ou diferentes algoritmos de aprendizado simbólico. Mais ainda, um conjunto de regras que constituem um classificador pode até ser um conjunto de regras criado pelo usuário. Isso permite construir indivíduos (classificadores) com uma grande diversidade, o que é desejável para o uso de AGs pois eles realizam uma busca global para construir o melhor classificador.

Com o objetivo de validar nossa proposta, projetamos um sistema computacional no qual definimos e implementamos os componentes do AG para evoluir classificadores simbólicos, tais como a estrutura de dados para representar os indivíduos (classificadores) e os operadores responsáveis pela evolução desses indivíduos. Esse sistema é explicado em detalhes neste trabalho, juntamente com algumas experiências realizadas utilizando diversas bases de dados e diferentes funções para avaliar os indivíduos. Os resultados experimentais permitem observar que, com algumas das funções de avaliação por nós propostas, é possível evoluir os classificadores iniciais para um classificador final que apresenta uma estimativa de erro de classificação menor que as taxas de erro dos classificadores iniciais.

O restante deste trabalho está organizado como segue: na Seção 2 são descritos alguns trabalhos relacionados à evolução de classificadores simbólicos utilizando AGs; na Seção 3 são apresentados conceitos para possibilitar a compreensão deste trabalho bem como a notação utilizada; na Seção 4 é descrito o AG proposto bem como os operadores do AG utilizados; na Seção 5 são descritos os experimentos realizados utilizando conjuntos de dados naturais, bem como os resultados obtidos; por fim, na Seção 6 encontram-se as conclusões deste trabalho e trabalhos futuros.

2 TRABALHOS RELACIONADOS

Para tratar o problema de manipulação de grandes bases de dados em mineração de dados utilizando algoritmos de aprendizado simbólico, podem ser utilizadas diferentes abordagens. Como mencionado, uma das abordagens é a abordagem de *ensembles*, na qual vários classificadores são induzidos sobre diversas amostras da base de dados disponível utilizando-se diferentes (ou eventualmente o mesmo) algoritmos de aprendizado simbólico. Após, as decisões do conjunto de classificadores são combinadas para classificar novos exemplos.

Em [3, 5], propomos diversos métodos de construção de *ensembles* de classificadores simbólicos, os quais foram implementados e avaliados utilizando bases de porte médio. Dado um novo exemplo a ser classificado, o *ensemble* de classificadores fornece, além da classificação desse exemplo, o conjunto de regras, as quais podem pertencer a diferentes classificadores, que cobrem esse exemplo. Esse conjunto de regras é utilizado para explicar esse exemplo. Devido ao fato desse conjunto de regras ter regras de diferentes classificadores, algumas dessas regras podem ser especializações de outras regras. Assim, em [4] propomos um método para resumir esse conjunto de regras antes de mostrá-las ao usuário. Os resultados obtidos utilizando os diversos métodos de construção de *ensembles* por nós propostos, bem como a explicação resumida da classificação de novos exemplos, foram encorajadores e mostraram a viabilidade dessa proposta.

Porém, uma desvantagem de *ensembles* é que o conjunto de todos os classificadores que o constituem deve ser armazenado. Mais ainda, cada um desses classificadores deve ser consultado para o *ensemble* determinar a classificação de novos exemplos. Isso motivou o desenvolvimento deste trabalho, no qual a idéia é construir um único classificador simbólico utilizando um conjunto de classificadores em vez de usar um *ensemble* de classificadores. Assim, propomos o uso de algoritmos genéticos para evoluir o conjunto de classificadores em um único classificador final. Dessa maneira, somente esse único classificador necessita ser armazenado e consultado para classificar novos exemplos.

Algoritmos genéticos têm sido amplamente utilizados em problemas de otimização. No caso de aprendizado, a idéia é evoluir tanto regras de conhecimento quanto classificadores com o objetivo de encontrar, respectivamente, a "melhor" regra ou a melhor hipótese que represente os dados disponíveis. Com essa finalidade, duas abordagens são propostas na literatura [9]: a abordagem Michigan, que evolui um conjunto de regras de conhecimento individuais em uma única regra de classificação; e a abordagem Pittsburgh, que evolui um conjunto de classificadores simbólicos em um único classificador simbólico. A abordagem Pittsburgh é utilizada para resolver o problema abordado neste trabalho. Como na abordagem Pittsburgh os indivíduos da população do AG são classificadores, ela deve levar em conta, no cômputo da função de avaliação dos indivíduos, a interação do conjunto de regras que constitui cada classificador. Diversos sistemas que utilizam a abordagem Pittsburgh têm sido propostos, tais como HDPDCS [15], EDRL [10] e EDRL-MD [11]. Um problema central é a codificação dos indivíduos da população manipulada pelo AG. Tal codificação pode gerar indivíduos sintaticamente longos, o que geralmente leva a modificações nas operações genéticas, dificultando o uso dos AGs para evoluir classificadores simbólicos. Nesse trabalho propomos uma codificação dos indivíduos que visa contemplar tais problemas.

3 CONCEITOS E NOTAÇÃO

No problema padrão de aprendizado supervisionado, ao algoritmo de aprendizado é dado um conjunto de exemplos de treinamento S com N exemplos $T_i, i = 1, ..., N$, escolhidos de um domínio $\mathcal{X}$ com uma distribuição D fixa, desconhecida e arbitrária, da forma $\{(\mathbf{x}_1,y_1),...,(\mathbf{x}_N,y_N)\}$ para alguma função desconhecida $y = f(\mathbf{x})$. Os $\mathbf{x}_i$ são tipicamente vetores da forma $(x_{i1},x_{i2},...,x_{iM})$, com valores discretos ou contínuos, onde x_{ij} refere-se ao valor do atributo j, denominado X_j, do exemplo T_i. Os valores y_i referem-se ao valor do atributo Y, freqüentemente denominado classe — Tabela 1.

	X_1	X_2	...	X_M	Y
T_1	x_{11}	x_{12}	...	x_{1M}	y_1
T_2	x_{21}	x_{22}	...	x_{2M}	y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
T_N	x_{N1}	x_{N2}	...	x_{NM}	y_N

Tabela 1: Conjunto de exemplos no formato atributo-valor

Em problemas de classificação, tratados neste trabalho, o atributo classe y_i é discreto, ou seja, $y_i \in \{C_1, C_2, ..., C_{N_{Cl}}\}$. A partir do conjunto de treinamento S, um *classificador* $\mathbf{h}$ é induzido. No caso de aprendizado simbólico proposicional [14], $\mathbf{h}$ pode ser transformado em um conjunto de regras de classificação **if-then** explicadas a seguir, *i.e.* $\mathbf{h} = \{R_1, R_2, ..., R_{N_R}\}$.

Um *complexo* é uma disjunção de conjunções de testes de atributos da forma X_i *op Valor*, onde X_i é o nome do atributo, *op* é um operador pertencente ao conjunto $\{=, \neq, <, \leq, >, \geq\}$ e *Valor* é um valor válido para o atributo X_i. Uma *regra proposicional* R apresenta a forma **if** B **then** H ou, simbolicamente, $B \rightarrow H$, onde H é a *cabeça*, ou a conclusão da regra R, e B é o *corpo*, ou condição de R. H e B são ambos complexos sem atributos em comum.

A *cobertura* de uma regra $R = B \rightarrow H$ é definida como segue: exemplos que satisfazem B (o corpo da regra) compõem o conjunto de cobertura de R; em outras palavras, esses exemplos são cobertos por R. Uma *regra de classificação* assume a forma **if** B **then** classe $= C_i$. Ou seja, a cabeça H de uma regra de classificação é classe $= C_i$, com $C_i \in \{C_1, ..., C_{N_{C_l}}\}$.

Dada uma regra e um conjunto de exemplos, uma maneira de avaliar essa regra é construindo sua matriz de contingência [12], mostrada na Tabela 2. Denotando a cardinalidade de um conjunto A como a, *i.e.* $a = |A|$, então b e h na Tabela 2 denotam, respectivamente, o número de exemplos pertencentes

aos conjuntos B and H, *i.e.* $b = |B|$ e $h = |H|$. De maneira análoga, $\overline{b} = |\overline{B}|$; $\overline{h} = |\overline{H}|$; $bh = |B \cap H|$; $\overline{b}h = |\overline{B} \cap H|$; $b\overline{h} = |B \cap \overline{H}|$; e $\overline{bh} = |\overline{B} \cap \overline{H}|$. A matriz de contingência de uma regra R permite o cálculo de diversas medidas de avaliação de regra, tais como acurácia e acurácia de Laplace, definidas pelas Equações 1 e 2 respectivamente, e outras [7, 12].

Tabela 2: Matriz de contingência para uma regra $B \rightarrow H$

	B	$\overline{B}$	
H	bh	$\overline{b}h$	h
$\overline{H}$	$b\overline{h}$	$\overline{bh}$	$\overline{h}$
	b	$\overline{b}$	N

$$Acc(R) = \frac{hb}{b} \tag{1}$$

$$Lacc(R) = \frac{bh + 1}{bh + b\overline{h} + N_{Cl}} \tag{2}$$

Além de avaliar as regras separadamente, também é necessário avaliar o classificador como um todo, levando assim em conta a interação entre as regras. Dado um conjunto de exemplos rotulados S, esse conjunto de exemplos deve ser subdividido em conjuntos a serem utilizados nas fases de treinamento e teste. Algumas vezes, também é necessário utilizar um conjunto de validação. O conjunto de treinamento é utilizado para induzir o modelo (hipótese), o conjunto de teste é utilizado para avaliar o modelo construído, e o conjunto de validação pode ser necessário em alguns casos para realizar ajustes no modelo construído pelo algoritmo de aprendizado. Como esses exemplos não são utilizados para construir o modelo mas são utilizados para seu "ajuste", esses exemplos não podem participar do conjunto de teste.

Matriz de Confusão: Para avaliar um classificador **h**, é necessário coletar informações das decisões tomadas pelo classificador no conjunto de teste. Para isso, constrói-se uma matriz bi-dimensional, cujas dimensões são denominadas *classe verdadeira* e *classe predita*. A essa matriz dá-se o nome de matriz de confusão, mostrada na Tabela 3, para N_{Cl} classes. Cada elemento $M(C_i, C_j)$ da matriz, definido pela Equação 3, indica o número de exemplos que pertencem à classe C_i e foram preditos como pertencentes à classe C_j. O número de predições corretas para cada classe são os apresentados na diagonal principal da matriz de confusão, ou seja, os valores associados a $M(C_i, C_i)$. Todos os outros elementos da matriz $M(C_i, C_j)$, para $i \neq j$, são referentes ao número de exemplos que foram erroneamente classificados em outra classe.

Classe Verdadeira	Predita C_1	Predita C_2	$\cdots$	Predita $C_{N_{Cl}}$
C_1	$M(C_1, C_1)$	$M(C_1, C_2)$	$\ldots$	$M(C_1, C_{N_{Cl}})$
C_2	$M(C_2, C_1)$	$M(C_2, C_2)$	$\ldots$	$M(C_2, C_{N_{Cl}})$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$C_{N_{Cl}}$	$M(C_{N_{Cl}}, C_1)$	$M(C_{N_{Cl}}, C_2)$	$\ldots$	$M(C_{N_{Cl}}, C_{N_{Cl}})$

Tabela 3: Matriz de confusão

$$M(C_i, C_j) = \sum_{\forall (\mathbf{x}, y) \in S | y = C_i} ||h(\mathbf{x}) = C_j|| \tag{3}$$

Dada a matriz de confusão, duas das medidas mais utilizadas para avaliar a performance de uma hipótese **h** são a acurácia e a taxa de erro, definidas pelas Equações 4 e 5 respectivamente. Outras medidas que também podem ser utilizadas para avaliar um classificador são sensibilidade ou *recall* (Equação 6), precisão (Equação 7) e F_1 (Equação 8)[1].

[1] A definição das medidas *Recall* e *Prec* para classes com mais de dois valores foi baseada na definição utilizada no KDD-Cup de 2005 — `http://www.acm.org/sigs/sigkdd/kdd2005/kddcup.html`.

$$Acc(\mathbf{h}) = \frac{\sum_{i=1}^{NCl} M(C_i, C_i)}{N} \tag{4}$$

$$Err(\mathbf{h}) = 1 - Acc(\mathbf{h}). \tag{5}$$

$$Recall(\mathbf{h}) = \frac{\sum_{i=1}^{N_{Cl}} M(C_i, C_i)}{\sum_{i=1}^{N_{Cl}} M(C_i, C_i) + \sum_{j=1}^{N_{Cl}} M(C_j, C_i)}. \tag{6}$$

$$Prec(\mathbf{h}) = \frac{\sum_{i=1}^{N_{Cl}} M(C_i, C_i)}{\sum_{i=1}^{N_{Cl}} M(C_i, C_i) + \sum_{j=1}^{N_{Cl}} M(C_i, C_j)}. \tag{7}$$

$$F_1(\mathbf{h}) = \frac{2 \times Prec(\mathbf{h}) \times Recall(\mathbf{h})}{Prec(\mathbf{h}) + Recall(\mathbf{h})}. \tag{8}$$

Algoritmos Genéticos: As seguintes definições estão baseadas em [13]. Um AG necessita de uma estrutura de dados que codifica uma solução para um problema que se deseja resolver, ou seja, uma representação para um ponto no espaço de busca. Tal estrutura é denominada *genoma* e tal ponto no espaço é denominado *cromossomo*. A união de um cromossomo com sua aptidão é denominada *indivíduo*. Um parâmetro codificado no cromossomo é denominado *gene*, e aos valores que um gene pode assumir dá-se o nome de *alelo*. O *genótipo* representa a informação contida no cromossomo ou genoma. Quando se decodifica o cromossomo, essa decodificação recebe o nome de *fenótipo*. Por exemplo, se um cromossomo codifica parâmetros de um conjunto de regras, o fenótipo é o conjunto de regras; se um cromossomo codifica os parâmetros de uma rede neural, o fenótipo é a rede neural. Dados um espaço de busca, uma função de aptidão, a qual determina a aptidão de cada indivíduo, ou seja, quão boa a solução por ele representada é para a resolução de um problema, e um conjunto de operadores genéticos (seleção, *crossover* e mutação), o algoritmo genético clássico segue os seguintes passos:

1. Gerar aleatoriamente uma população de n cromossomos.
2. Para cada cromossomo x da população, calcular a função de aptidão $f(x)$ (população avaliada).
3. Repetir os seguintes passos até que n filhos tenham sido criados:
   (a) Selecionar um par de cromossomos da população corrente. A probabilidade de seleção do par de cromossomos é dada pela função de aptidão.
   (b) Aplicar a operação de *crossover* sobre os cromossomos selecionados, produzindo um único filho, resultante da união de uma parte de um dos cromossomos selecionados com outra parte do outro cromossomo.
   (c) Aplicar a operação de mutação em cada locus do filho criado e colocar o cromossomo resultante na nova população. A operação de mutação tipicamente substitui o valor de um locus por outro valor.
4. Substituir a população corrente pela nova população.
5. Voltar ao passo 2 se o(s) critério(s) de parada não for(em) satisfeito(s).

Podem existir diversos critérios de parada, tais como alcançar um número pré-fixado de gerações ou a função de aptidão permanecer constante. *Geração* é o nome dado a cada iteração do algoritmo descrito. O conjunto completo de iterações é denominado *execução*.

AGs é um dentre uma classe de paradigmas de Algoritmos Evolutivos para simular a evolução utilizando os conceitos de evolução de Darwin para, de forma iterativa, gerar soluções apropriadas (indivíduos) incrementalmente em um ambiente estático ou que muda dinamicamente [9].

4 O ALGORITMO GENÉTICO PROPOSTO

Devido à dificuldade de se aplicar o algoritmo genético clássico, descrito anteriormente, a todo tipo de problema, várias implementações distintas foram criadas para problemas particulares. Por outro lado, o principal objetivo de aprendizado de máquina envolve construir programas de computador capazes de construir novos conhecimentos ou melhorar conhecimento previamente construído, utilizando como informação de entrada observações (casos ou exemplos) do mundo. Há um grande interesse em aplicar técnicas de computação evolutiva em aprendizado de máquina devido à idéia atrativa de cromossomos, representando conhecimento, serem tratados como dados (casos ou exemplos) a serem manipulados por operadores genéticos, e, ao mesmo tempo, serem códigos executáveis para serem utilizados na realização de alguma tarefa.

O AG aqui proposto segue os passos do AG clássico, descrito na seção anterior, utilizando a abordagem Pittsburgh, descrita na Seção 2. O que varia na nossa proposta são os componentes do AG. A vantagem do algoritmo genético por nós proposto, em relação a outros AGs descritos na literatura, está relacionado com a codificação dos indivíduos, e com a população inicial do AG. Propomos utilizar, além das operações de crossover e mutação usuais, diferentes funções de avaliação. A seguir, são descritos os componentes do AG proposto neste trabalho, e as estruturas utilizadas para a sua implementação.

População Inicial, Inicialização do AG e Codificação dos Indivíduos: No AG proposto, cada gene de um indivíduo (classificador) é uma regra, e cada indivíduo é, portanto, uma seqüência (conjunto) de regras. Na população inicial do AG proposto, podem participar:

1. classificadores induzidos por diversos algoritmos de aprendizado simbólico, tais como $\mathcal{CN}2$ [8], $\mathcal{C}4.5$ e $\mathcal{C}4.5rules$ [19] ou outros algoritmos;
2. classificadores constituídos por conjunto de regras fornecidos pelo usuário;
3. classificadores construídos pelo sistema computacional que implementa o AG, descrito adiante. Para construir tais classificadores, o sistema computacional seleciona regras de uma base de regras. Essa base é formada por regras que participam dos classificadores descritos nos itens anteriores.

A idéia de utilizar essa base de regras (item 3) tem como objetivo permitir ao usuário a definição de novas regras, as quais têm a possibilidade de participar do classificador final evoluído pelo AG. Observe que os classificadores iniciais constituídos no item 3 podem ser muito pouco precisos. Entretanto, algumas das regras que constituem esses classificadores pode eventualmente participar no classificador final. Todas as regras são transformadas para a sintaxe padrão $\mathcal{PBM}$ [18], descrita mais adiante, e armazenadas na base de regras, ilustrada na Figura 1. Cada uma dessas regras possui um identificador único.

Antes de executar o AG, o usuário deve fornecer ao sistema implementado alguns parâmetros, dentre eles o tamanho inicial dos indivíduos N_R e o número de indivíduos N_{ind} na população. Para formar a população inicial, o sistema seleciona, para construir cada indivíduo, N_R regras diferentes da base de regras. Cada indivíduo é então codificado como sendo um vetor com identificadores de regras pertencentes à base de regras, ilustrado na Figura 2. Também, como opção, o usuário pode requerer que os classificadores fornecidos como entrada para compor a base de regras do AG (itens 1 e 2) também participem da população inicial do AG. Assim, são formados N_{ind} indivíduos.

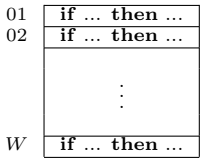

Figura 1: Base de regras na sintaxe padrão $\mathcal{PBM}$.

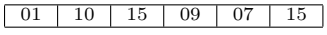

Figura 2: Exemplo de codificação de um indivíduo

Operadores Genéticos: São dois os operadores genéticos utilizados no AG proposto: *crossover* e mutação. O operador de *crossover* utilizado é o *crossover* assimétrico. Em um *crossover* assimétrico, a cada indivíduo da população corrente é associado aleatoriamente um valor entre 0 e 1. Os indivíduos, cujo valor associado é menor que a probabilidade de *crossover* p_c, fornecido pelo usuário, são selecionados para a operação de

crossover. Para cada par de indivíduos "pais"são escolhidas duas posições, uma em cada indivíduo. Os indivíduos pais são divididos em dois segmentos conforme as posições escolhidas. Por fim, são criados os indivíduos filhos por meio da composição dos segmentos dos indivíduos pai. A operação de mutação é utilizada de maneira usual: para aplicar a operação de mutação, seleciona-se um gene de um indivíduo (aleatoriamente) e troca-se a regra selecionada por outra selecionada da base de regras disponível. O número de genes selecionados é dado pela probabilidade de mutação p_m, fornecida pelo usuário.

Funções de Avaliação: Para avaliar um conjunto de regras, deve-se avaliar o comportamento do conjunto de regras (classificador) sobre um conjunto de exemplos de teste. Assim, pode-se avaliar o comportamento do conjunto de regras tanto como uma "caixa preta" quanto fazer uma avaliação do conjunto de regras juntamente com o comportamento das regras individuais. Para avaliar os classificadores como sendo uma caixa preta, pode-se utilizar, por exemplo, as medidas de acurácia, precisão e F_1, definidas na Seção 3. Neste trabalho propomos as seguintes duas medidas — Equações 9 e 10 — denominadas HQ (*Hypothesis Quality*) para avaliar o desempenho do classificador.

$$HQ_{Acc}(\mathbf{h}) = Acc(\mathbf{h}) \tag{9}$$

$$HQ_{F1}(\mathbf{h}) = F1(\mathbf{h}) \tag{10}$$

Deve ser observado que, por se tratar de classificadores simbólicos, podem ser utilizadas diferentes maneiras para determinar a classificação de um novo exemplo, *i.e.*, dado um exemplo $\mathbf{x}$ e um classificador $\mathbf{h} = \{R_1, ..., R_{N_R}\}$, podem ser utilizados diversos métodos para classificar $\mathbf{x}$. Neste trabalho, propomos diversas maneiras de classificar $\mathbf{x}$, as quais podem ser enquadradas em dois grandes grupos, denominados SR (*Simple Rule classification method*) e MR (*Multiple Rule classification method*). No primeiro grupo, $\mathbf{x}$ é classificado utilizando a "melhor" regra do classificador, segundo uma medida de qualidade de regras. Atualmente, as seguintes duas medidas encontram-se implementadas no sistema que implementa o AG proposto:

1. SR_{Acc}: É utilizada a melhor regra segundo a medida de avaliação de regra precisão ($Acc(R)$ — Equação 1) para classificar $\mathbf{x}$;
2. SR_{Lacc}: É utilizada a melhor regra segundo a medida de avaliação de regra precisão de Laplace ($Lacc(R)$ — Equação 2) para classificar $\mathbf{x}$.

No segundo grupo, MR, todo o conjunto de regras que constitui o classificador é utilizado para classificar $\mathbf{x}$. Os seguintes dois procedimentos são utilizados:

1. MR_{Acc}: São utilizadas todas as regras que cobrem $\mathbf{x}$ para encontrar sua classificação. Para determinar a classe de $\mathbf{x}$, para cada classe $C_v \in \{C_1, ..., C_{N_{Cl}}\}$ é somado o valor da medida de precisão ($Acc(R)$) para as regras que classificam o exemplo na classe em questão. A classe com maior valor total é a classificação de $\mathbf{x}$.
2. MR_{Lacc}: São utilizadas todas as regras que cobrem $\mathbf{x}$ para encontrar sua classificação. Para determinar a classe de $\mathbf{x}$, para cada classe $C_v \in \{C_1, ..., C_{N_{Cl}}\}$ é somado o valor da medida de precisão de laplace ($Lacc(R)$) para as regras que classificam o exemplo na classe em questão. A classe com maior valor total é a classificação de $\mathbf{x}$.

Critério de Parada: O critério de parada atualmente implementado é dado pelo número máximo de gerações, o qual deve ser dado como parâmetro pelo usuário, que o AG deve executar. Como saída, o AG oferece então o melhor indivíduo da última geração executada. Outros critérios de parada que levem também em consideração o comportamento do AG serão implementados futuramente, com o objetivo de analisar, entre outros, a velocidade de convergência do algoritmo proposto.

Pós-processamento do Indivíduo Resultante: Após o AG fornecer como saída o melhor indivíduo (classificador) simbólico por ele evoluído, o sistema proposto realiza um pós-processamento desse classificador. Isso é realizado da seguinte maneira: dado o conjunto de exemplos de treinamento S_{tr}, antes de iniciar a execução do AG, são retirados 10% dos exemplos desse conjunto S_{tr}, formando assim o conjunto de validação

S_{val}, tal que a proporção de exemplos de S_{val} em cada classe é semelhante à do conjunto S_{tr}, sendo $S_{tr} = S_{val} \cup S'_{tr}$. Assim, o AG é executado utilizando o conjunto S'_{tr}, *i.e.*, 90% dos exemplos do conjunto de treinamento S_{tr} fornecido como entrada ao AG. O conjunto S'_{tr} é utilizado a cada iteração do AG para cálculo da função de avaliação selecionada pelo usuário para ser utilizada. Ao final da execução, o melhor indivíduo (classificador) encontrado é pós-processado utilizando todo o conjunto de exemplos de treinamento original S_{tr}, *i.e.*, 90% dos exemplos de S_{tr}, pertencentes a S'_{tr}, mais 10% dos exemplos de S_{tr}, pertencentes ao conjunto de validação S_{val}. O critério de pós-processamento utilizado consiste em retirar desse classificador todas as regras que não cobrem nenhum exemplo em S_{tr}.

Com o objetivo de avaliar nossa proposta, foi implementado um sistema computacional denominado *Genetic Algorithms for Evolving Rule Sets Environment* (GAERE), integrado ao ambiente computacional DISCOVER. O ambiente DISCOVER tem como principal objetivo integrar e padronizar os diversos projetos desenvolvidos no Laboratório de Inteligência Computacional — LABIC — do ICMC-USP, relacionados com pré-processamento de conjuntos de dados, aquisição automática de conhecimento e avaliação de conhecimento [17]. Na maioria desses projetos, incluindo o projeto GAERE, diversas tarefas tais como transformação de dados e formatos, execução de algoritmos, medições, entre outras, devem ser executadas diversas vezes. Assim, muitas ferramentas foram e estão sendo implementadas na linguagem de programação *Perl* [16] no DISCOVER, como bibliotecas de classes, para automatizar parcial ou integralmente algumas dessas tarefas.

O ambiente DISCOVER oferece vantagens em relação a outros sistemas com objetivos semelhantes, pois permite a visão unificada que os formatos baseados em padrões proporcionam ao pesquisador (desenvolvedor) de novos componentes. Os padrões de representação foram sendo definidos por área. Em [18] é proposta uma sintaxe padrão para representação de conhecimento de diversos indutores simbólicos denominada $\mathcal{PBM}$. A sintaxe padrão $\mathcal{PBM}$ é implementada no DISCOVER como uma biblioteca de classes. Essa biblioteca converte os classificadores induzidos na linguagem de representação de conceitos dos principais algoritmos de AM simbólicos, tais como $\mathcal{CN}2$ [8], $\mathcal{C}4.5$ e $\mathcal{C}4.5rules$ [19] para a sintaxe padrão $\mathcal{PBM}$. Nessa sintaxe padrão, cada regra que compõe o classificador simbólico é reescrita na forma *if* <condição> *then* <class = C_i >, seguindo uma gramática definida em [18].

Para a representação de dados foi proposta uma sintaxe padrão, denominada DSX — *Discover Dataset Sintax*, a qual permite a utilização da biblioteca de classes DOL [2], para converter os arquivos de dados para a sintaxe utilizada por diversos sistemas de aprendizado simbólico, tais como $\mathcal{CN}2$ [8], $\mathcal{C}4.5$, $\mathcal{C}4.5rules$ [19], entre outros. O sistema GAERE, no qual está implementado o algoritmo genético proposto, foi desenvolvido como uma biblioteca de classes do DISCOVER utilizando várias de suas funcionalidades.

5 EXPERIMENTOS E RESULTADOS

Com o objetivo de avaliar o AG proposto para evoluir classificadores simbólicos, foram realizados diversos experimentos utilizando 4 (quatro) conjuntos de dados disponíveis na UCI [6] — autos, heart, ionosphere e vehicle. Na Tabela 4, são mostradas algumas características desses conjuntos de dados: número de exemplos (# Ex.), número de atributos (contínuos, discretos) (# Atr.) e número de valores no atributo classe (# Classes). Deve ser observado que somente o conjunto de dados autos possui valores desconhecidos, somente o conjunto de dados ionosphere possui um exemplo duplicado, e a distribuição dos dados nas classes é uniforme.

Conj. Dados	# Ex.	# Atr. (cont.,disc.)	# Classes
autos	205	25 (15,10)	2
heart	270	13 (5,8)	2
vehicle	846	18 (18,0)	4
ionosphere	351	33 (33,0)	2

Tabela 4: Sumário das características dos conjuntos de dados utilizados

Diversos experimentos foram realizados utilizando o sistema GAERE, variando a função de avaliação. Inicialmente, foram induzidos 3 classificadores utilizando os algoritmos de aprendizado simbólico $\mathcal{CN}2$ [8], $\mathcal{C}4.5$ e $\mathcal{C}4.5rules$ [19], denominados $\mathbf{h}_{\mathcal{CN}2}$, $\mathbf{h}_{\mathcal{C}4.5}$ e $\mathbf{h}_{\mathcal{C}4.5rules}$, respectivamente. Após, as regras desses três classificadores foram utilizados para compor a base de regras do AG. Como parâmetros do AG, foram utilizados os seguintes: número de ciclos, utilizado como critério de parada do AG: 10; número de regras N_R pertencentes aos indivíduos iniciais: foi calculada a média do número de regras dos classificadores $\mathbf{h}_{\mathcal{CN}2}$, $\mathbf{h}_{\mathcal{C}4.5}$ e $\mathbf{h}_{\mathcal{C}4.5rules}$, a qual foi utilizada como número de regras dos classificadores iniciais; tamanho da população,

i.e., número de indivíduos a serem evoluídos: 15; probabilidade de *crossover*: 0.40; probabilidade de mutação: 0.01. Na realidade, esses parâmetros são os parâmetros *default* do AG implementado no GAERE.

Para inicializar a população inicial do AG, como o número de classificadores é 15, nos experimentos realizados foram utilizados os 3 classificadores $\mathbf{h}_{\mathcal{CN}2}$, $\mathbf{h}_{\mathcal{C}4.5}$ e $\mathbf{h}_{\mathcal{C}4.5rules}$, juntamente com 12 classificadores construídos selecionando-se da base de regras N_R regras distintas para cada classificador.

Na Tabela 5, são mostrados os resultados obtidos nos experimentos. Nessa tabela, na primeira linha estão os erros majoritários de cada conjunto de dados. Nas linhas 2, 3 e 4, estão as taxas de erro estimadas dos classificadores induzidos utilizando os algoritmos $\mathcal{CN}2$, $\mathcal{C}4.5$ e $\mathcal{C}4.5rules$ respectivamente. Nas últimas quatro linhas dessa tabela, estão as taxas de erro estimadas para a evolução dos classificadores utilizando o AG proposto com os parâmetros *default* e variando a função de avaliação. Nessa tabela, $MR_{Acc}HQ_{Acc}$ significa que foram utilizados na função de avaliação do AG o método de classificação MR_{Acc} e a medida de desempenho do classificador HQ_{Acc}; $MR_{Acc}HQ_{F1}$ significa que foram utilizados o método de classificação MR_{Acc} e a medida de desempenho do classificador HQ_{F1}; e assim sucessivamente.

Tabela 5: Resultados obtidos com o AG proposto variando a função de avaliação. Utilizando o teste t com 95% de confiança, o símbolo "+" indica que o classificador evoluído é melhor que o melhor dentre os classificadores iniciais, o símbolo "-" indica que o classificador evoluído é pior, e o símbolo "o" indica que não houve diferença significativa.

	autos	heart	ionosphere	vehicle
EM	44,88%	44,44%	35,90%	74,23%
$\mathcal{CN}2$	10,29% (1,87%)	22,96% (2,46%)	9,09% (1,86%)	39,83% (1,57%)
$\mathcal{C}4.5$	9,31% (1,71%)	29,26% (1,78%)	10,56% (1,76%)	27,31% (1,37%)
$\mathcal{C}4.5rules$	8,71% (1,73%)	20,37% (1,27%)	9,96%(1,48%)	25,77% (0,85%)
$MR_{Acc}HQ_{Acc}$	8,17% (4,60%) o	13,33% (2,54%) +	6,56% (1,27%) o	20,87% (0,66%) +
$MR_{Acc}HQ_{F1}$	4,36% (1,95%) +	10,74% (2,10%) +	6,57% (1,21%) o	24,42% (1,51%) o
$MR_{Lacc}HQ_{Acc}$	5,33% (1,83%) o	11,11% (1,56%) +	6,83% (1,21%) o	21,27% (0,85%) +
$MR_{Lacc}HQ_{F1}$	3,86% (1,39%) +	12,96% (1,77%) +	7,11% (1,28%) o	24,93% (1,53%) o
$SR_{Acc}HQ_{Acc}$	15,93% (4,23%) -	27,04% (3,62%) -	18,81% (2,34%) -	51,42% (2,37%) -
$SR_{Acc}HQ_{F1}$	18,98% (3,93%) -	26,67% (3,06%) -	20,24% (3,18%) -	45,73% (3,17%) -
$SR_{Lacc}HQ_{Acc}$	14,55% (3,54%) -	20,37% (2,23%) o	11,97% (2,03%) o	35,13% (2,64%) -
$SR_{Lacc}HQ_{F1}$	18,93% (3,64%) -	20,00% (1,67%) o	14,24% (2,88%) -	36,16% (1,41%) -

Para estimar as taxas de erro mostradas na Tabela 5, foi utilizada a técnica de 10-*fold cross-validation*. Para os classificadores iniciais induzidos por $\mathcal{CN}2$, $\mathcal{C}4.5$ e $\mathcal{C}4.5rules$, foi utilizada a maneira usual de execução do 10-*fold cross-validation* [1]. Para calcular a estimativa do erro dos classificadores finais evoluídos pelo AG, também foi utilizado o 10-*fold cross-validation*. Entretanto, essa técnica foi adaptada para o problema de estimar a taxa de erro com a execução do AG proposto. Nessa adaptação, na primeira iteração (it_1) do 10-$fold$, é separado o conjunto de treinamento S_{tr_1} e teste S_{te_1} da maneira usual em 10-*fold cross-validation*. Do conjunto de treinamento, são extraídos 10% dos exemplos de maneira estratificada, *i.e.*, respeitando a distribuição dos dados no atributo classe, para formar o conjunto de validação S_{val_1}. O conjunto S_{val_1} é separado para a fase de pós-processamento. Sobre o conjunto de treinamento restante S'_{tr_1}, onde $S'_{tr_1} = S_{tr_1} - S_{val_1}$, são induzidos os classificadores $\mathbf{h}_{\mathcal{CN}2_1}$, $\mathbf{h}_{\mathcal{C}4.5_1}$ e $\mathbf{h}_{\mathcal{C}4.5rules_1}$, utilizando respectivamente os algoritmos de aprendizado $\mathcal{CN}2$, $\mathcal{C}4.5$ e $\mathcal{C}4.5rules$. Esses três classificadores e o conjunto de exemplos S'_{tr_1} são dados de entrada do AG, bem como a função de avaliação a ser utilizada.

O AG é então executado: lembrando que são 15 os indivíduos que formam a população a ser evoluída, são carregadas todas as regras dos classificadores $\mathbf{h}_{\mathcal{CN}2_1}$, $\mathbf{h}_{\mathcal{C}4.5_1}$ e $\mathbf{h}_{\mathcal{C}4.5rules_1}$, tais classificadores são considerados como 3 dos classificadores da população inicial, e mais 12 indivíduos são criados, cada um com diferentes regras da base de regras. Após as 10 gerações do AG, o classificador resultante do processo (o classificador evoluído) $\mathbf{h}_1$ é pós-processado, *i.e.*, são retiradas as regras que não cobrem nenhum exemplo no conjunto $S_{tr_1} = S'_{tr_1} + S_{val_1}$. Do pós-processamento resulta um conjunto de regras, que forma a hipótese resultante (evoluída) $\mathbf{h}^*_1$. Sobre $\mathbf{h}^*_1$ é calculada a taxa de erro utilizando o conjunto de teste S_{te_1}, sendo essa a taxa de erro da primeira iteração ε_1. O processo é repetido 10 vezes, obtendo-se assim 10 taxas de erro $\varepsilon_1, ..., \varepsilon_{10}$ nas 10 iterações do 10-*fold cross-validation*. Assim, estima-se a média e o erro padrão sobre as 10 taxas de erro $\varepsilon_1, ..., \varepsilon_{10}$.

Para verificar se o resultado do classificador final é significativamente melhor (ou pior) que o classificador

inicial que apresenta menor taxa de erro, foi realizado o teste t. Para cada resultado com o AG na Tabela 5, é colocado o símbolo "+" para indicar que o classificador final é melhor que o melhor dos classificadores iniciais, pois a diferença entre a taxa de erro do classificador final evoluído pelo AG e a menor taxa de erro dentre os classificadores iniciais é significativa; o símbolo "-" para indicar que o classificador final é pior que o melhor dos classificadores iniciais, pois a diferença entre a taxa de erro do classificador final e a menor taxa de erro dentre os classificadores iniciais é significativa; e é colocado o símbolo "o" para indicar que não há diferença significativa entre as taxas de erro, segundo o teste t com 95% de confiança.

Para facilitar a visualização dos resultados obtidos, foi construído um gráfico, exibido na Figura 3. Nesse gráfico, os valores do eixo das abscissas são os valores estimados das taxas de erro dos classificadores finais encontrados pelo AG, e os valores do eixo das ordenadas referem-se ao mínimo valor encontrado dentre as taxas de erro estimadas dos classificadores iniciais fornecidos ao AG.

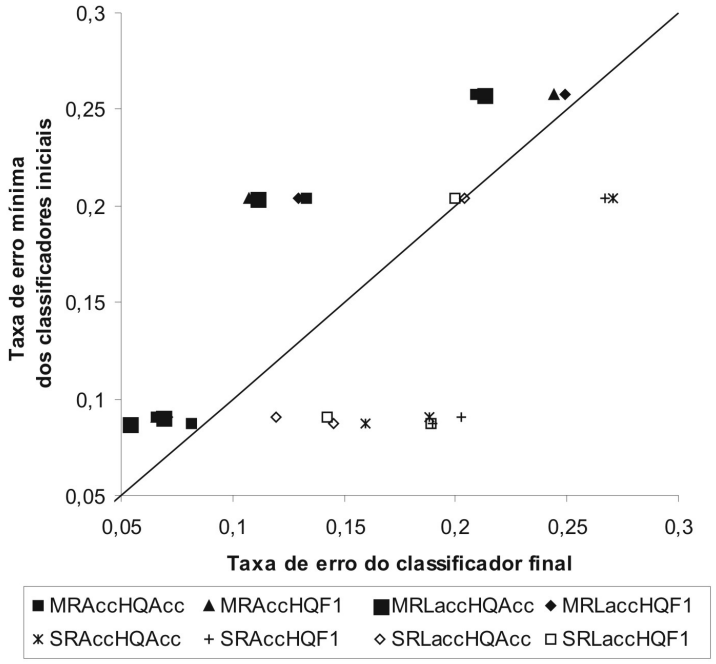

Figura 3: Resultados obtidos nos conjuntos de dados utilizados. Os pontos "acima" da reta indicam que a taxa de erro estimada do classificador final é menor que a taxa de erro do melhor classificador inicial, e os pontos "abaixo" da reta indicam que a taxa de erro estimada do classificador final é maior que a taxa de erro do melhor classificador inicial.

Tanto na Tabela 5 quanto no gráfico exibido na Figura 3, pode ser observado que os métodos que utilizam somente uma regra para classificar os exemplos, independente do método de avaliação do classificador, piora a taxa de erro quando comparada aos classificadores iniciais. Ainda, segundo o teste t, na maioria dos experimentos utilizando os métodos que utilizam somente uma regra para classificar os exemplos na função de avaliação, a taxa de erro do classificador final é pior que a menor taxa de erro dos classificadores iniciais com 95% de confiança nos experimentos realizados. Já na maioria dos experimentos utilizando os métodos que utilizam todas as regras que cobrem o exemplo para classificá-lo, a taxa de erro do classificador final é melhor que a menor taxa de erro dos classificadores iniciais com 95% de confiança nos experimentos realizados.

Resumindo os resultados, utilizando como função de avaliação as medidas do grupo MR (*Multiple Rule classification method*), para o conjunto de dados ionosphere pode ser observado que não houve melhora significativa para nenhuma das quatro medidas, apesar de as taxas de erro utilizando os métodos MR serem menores que as taxas de erro dos classificadores iniciais. Para o conjunto de dados autos, houve melhora significativa quando a medida de avaliação da hipótese utilizada foi HQ_{F1}, enquanto que para o conjunto de

dados vehicle houve melhora significativa quando a medida de avaliação da hipótese utilizada foi HQ_{Acc}.

6 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foi proposto um algoritmo genético para evoluir classificadores simbólicos utilizando a abordagem Pittsburgh para evoluir classificadores iniciais em um único classificador final. A vantagem do AG em relação a *ensembles* de classificadores é que somente o classificador final necessita ser armazenado para realizar a classificação de novos exemplos, enquanto que, utilizando *ensembles* de classificadores, todos os classificadores que compõem o *ensemble* devem ser armazenados para a classificação de exemplos futuros.

Para o AG proposto, foram propostas 8 (oito) diferentes funções de avaliação, as quais podem ser divididas em dois grandes grupos: SR (*Single Rule classification method*) e MR (*Multiple Rule classification method*). Para compor os indivíduos (classificadores) iniciais bem como para executar a operação de mutação, na qual é necessário substituir uma regra de um classificador por outra regra diferente, é utilizada uma base de regras, a qual pode ser composta por regras induzidas de algoritmos de aprendizado simbólico ou por regras criadas pelo próprio usuário. As vantagens do AG por nós proposto em relação a outros AGs encontrados na literatura estão na codificação dos indivíduos, pois todas as regras que compõem os indivíduos são escritas em uma sintaxe padrão de regras — a sintaxe $\mathcal{PBM}$ —, e na liberdade do usuário inicializar a base de regras com regras tanto induzidas por algoritmos de aprendizado simbólico quanto por regras por ele criadas.

O AG proposto foi implementado no sistema computacional GAERE (*Genetic Algorithms for Evaluation Rulesets Environment*). Para testar o AG, foram utilizados 4 (quatro) conjuntos de dados da UCI, e foram utilizadas as oito funções de avaliação propostas. Utilizando as funções de avaliação da família SR, não houve melhora no classificador evoluído, muito provavelmente devido ao fato de somente uma regra ser utilizada para classificar novos exemplos. Entretanto, utilizando as funções de avaliação da família MR, houve melhora na taxa de erro dos classificadores evoluídos em relação ao melhor classificador dentre os classificadores iniciais (induzidos com os algoritmos $\mathcal{CN}2$ [8], $\mathcal{C}4.5$ e $\mathcal{C}4.5rules$ [19]) em 8 dos 16 experimentos realizados, segundo o teste t, com 95% de confiança.

Além de avaliar o desempenho dos classificadores em relação a seu poder de predição, ou seja, taxa de erro de classificação, também é necessário avaliar os classificadores em relação à qualidade das regras que compõem o classificador. Como trabalhos futuros, serão avaliadas as regras que compõem os classificadores evoluídos com o AG, com auxílio de especialistas do domínio.

Referências

[1] J. A. Baranauskas and M. C. Monard. Reviewing some machine learning concepts and methods. Technical Report 102, ICMC/USP, Fev 2000. `ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_102.ps.zip`.

[2] G. E. A. P. A. Batista and M. C. Monard. Descrição da arquitetura e do projeto do ambiente computacional Discover Learning Environment - DLE. Technical Report 187, ICMC/USP, 2003. `ftp://ftp.icmc.usp.br/pub/BIBLIOTECA/rel_tec/RT_187.PDF`.

[3] F. C. Bernardini and M. C. Monard. Methods for constructing symbolic ensembles from symbolic classifiers. In *Congress of Logic Applied to Technology — LAPTEC 2005. Frontiers in Artificial Intelligence and Applications*, volume 132, pages 161–168, Hyogo, Japan, 2005. Netherlands: IOS Press.

[4] F. C. Bernardini and M. C. Monard. Uma proposta para a construção de ensembles simbólicos que explicam suas decisões. In *Conf. Latinoamericana de Informatica — CLEI 2005*, volume 1, pages 151–162, Cali, Colômbia, 2005.

[5] F. C. Bernardini, M. C. Monard, and R. C. Prati. Constructing ensembles of symbolic classifiers. In *International Conference on Hybrid Intelligent Systems — HIS 2005*, volume 1, pages 315–320, Rio de Janeiro, 2005. California: IEEE Computer Society.

[6] C. Blake, E. Keogh, and C. J. Merz. UCI repository of machine learning databases, 1998. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[7] P. Clark and R. Boswell. Rule induction with $\mathcal{CN}2$: Some recent improvements. In Y. Kodratoff, editor, *Proceedings of the 5th European Working Session on Learning (EWSL 91)*, pages 151–163, 1991.

[8] P. Clark and T. Niblett. The $\mathcal{CN}2$ induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

[9] A. A. Freitas. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer Verlag, 2002.

[10] W. Kwedlo and M. Kretowski. Discovery of decision rules from databases: an evolutionary approach. In *Proc. 2nd Eur. Symp. on Princ. Knowledge Discovery in Databases(PKDD-98). LNAI*, volume 1510, pages 371–378. Springer Verlag, 1998.

[11] W. Kwedlo and M. Kretowski. An evolutionary algorithm for cost-sensitive decision rule learning. In *12th Eur. Conf. on Machine Learning — ECML'2001. LNAI*, volume 2167, pages 288–299. Springer Verlag, 2001.

[12] N. Lavrac, P. Flach, and B. Zupan. Rule evaluation measures: a unifying view. In *Proc. 9th Int. Workshop on Inductive Logic Programming. LNAI*, volume 1634, pages 74–185. Springer Verlag, 1999.

[13] M. Mitchell. *An Introduction to Genetic Algorithms*. The MIT Press, 1997.

[14] M. C. Monard and J. A. Baranauskas. *Indução de Regras e Árvores de Decisão*, chapter 5, pages 115–140. In [20], 2003.

[15] M. Pei, E. D. Goodman, and W. F. Punch. Pattern discovery from data using genetic algorithms. In *Proc. 1st Pacific-Asia Conf. Knowledge Discovery & Data Mining (PAKDD-97)*, Fev 1997.

[16] PERL. *Programming in PERL*. Morgan Kaufmann Publishers, Inc., 1999.

[17] R. C. Prati. O framework de integração do sistema DISCOVER, 2003. Dissertação de Mestrado, ICMC/USP. `http://www.teses.usp.br/teses/disponiveis/55/55134/tde-20082003-152116/`.

[18] R. C. Prati, J. A. Baranauskas, and M. C. Monard. Padronização da sintaxe e informações sobre regras induzidas a partir de algoritmos de aprendizado de máquina simbólico. *Revista Eletrônica de Iniciação Científica*, 2(3), 2002. `http://www.sbc.org.br/reic/edicoes/2002e3`.

[19] J. R. Quinlan. *C4.5 Programs for Machine Learning*. Morgan Kaufmann Publishers, 1988.

[20] S. O. Rezende. *Sistemas Inteligentes: Fundamentos e Aplicações*. Ed. Manole, Brasil, 2003.