INDUÇÃO DE ÁRVORES DE PADRÕES FUZZY ATRAVÉS DA PROGRAMAÇÃO GENÉTICA CARTESIANA

ANDERSON R. DOS SANTOS, CESAR EDUARDO R.F. DE ALMEIDA, CARLOS AUGUSTO R. SOARES, JORGE LUÍS M. DO AMARAL

Laboratório de Redes Industriais e Sistemas de Automação, Faculdade de Engenharia,Universidade do Estado do Rio de Janeiro

E-mails: andersonrds_rj@hotmail.com, cesarlmd63@gmail.com, caugusto.rio@gmail.com, jamaral@uerj.br

Abstract— This paper describes a system for induction of fuzzy classifiers based on Fuzzy Pattern Trees instead of the traditional fuzzy based rules. FPT which is a hierarchical tree-based model, having as internal nodes, fuzzy logical operators and the leaves are composed by the combination of fuzzy terms and the input attributes. The classifier was obtained by creating a tree for each class. Therefore each tree will be a "logic class description" which allows the interpretation of the results. The learning method originally designed for generating a Fuzzy Pattern Trees was replaced by Cartesian Genetic Programming in order to provide a better exploration of the search space. The Fuzzy Pattern Trees classifier was compared against Support Vector Machines, K Nearest Neighbors and Random Forests on several datasets from the UCI Machine Learning Repository and it presented competitive results. It was also compared to Fuzzy Pattern trees generated by the former learning method and presented comparable results with smaller trees.

Keywords— Machine Learning, Fuzzy Pattern Trees, Cartesian Genetic Programming, Classification, interpretability.

Resumo— Este artigo descreve um sistema para indução de classificadores *fuzzy* baseados em Árvores de Padrões Fuzzy diferente dos tradicionais sistemas *fuzzy* baseados em regras. Árvores de Padrões Fuzzy é um modelo hierárquico baseado em árvores, tendo como nós internos, operadores lógicos *fuzzy* e suas folhas são compostas pela combinação de termos *fuzzy* e atributos de entrada. O classificador foi obtido criando uma árvore para cada classe. Cada árvore será uma "Descrição lógica da classe" permitindo a interpretação do resultado. O método de aprendizado desenvolvido originalmente para gerar as APF foi substituído pela Programação Genética Cartesiana, de modo a proporcionar uma melhor exploração do espaço de buscas. O classificador Árvores de Padrões Fuzzy foi comparado com Support Vector Machines, K Nearest Neighbors e Random Forests, em diversas bases de dados do UCI Machine Learning Repository e apresentou resultados competitivos. Também foi comparado com árvores geradas pelos métodos de aprendizado originais, apresentando resultados competitivos e árvores menores.

Palavras-chave— Aprendizado de Maquinas, Árvores de Padrões Fuzzy, Programação Genética Cartesiana, Classificação, Interpretabilidade.

1 Introdução

Hoje, devido a grandes avanços tecnológicos, é possível coletar uma quantidade enorme de informações em diversas áreas de *expertise*. Entretanto, permanece o desafio da análise desses dados para se extrair um conhecimento útil. A análise de uma quantidade tão grande de dados não pode ser feita por seres humanos em um tempo razoável. Portanto, esta tarefa atrai um interesse considerável no estudo de modelos que podem aprender a partir de um conjunto de dados (Witten e Frank, 2005). A indução destes modelos pode ser feita de uma maneira automática, através da utilização de diferentes métodos, tais como: Redes Neurais Artificiais, Métodos Bayesianos, Modelos Gráficos, Árvores de Decisão, entre outros. No entanto, quando se deseja entender "como" o modelo induzido tomou a sua decisão, ou seja, se quisermos ter uma explicação sobre como a decisão é feita, então os modelos gerados por abordagens simbólicas, como por exemplo, sistemas *fuzzy* baseados em regras se tornam mais atraentes, pois eles podem gerar modelos interpretáveis.

Alguns dos métodos com aplicações mais bem sucedidas na síntese de modelos interpretáveis são baseados na teoria dos conjuntos *Fuzzy*(Cordón, 2011; Herrera, 2008). Pois sistemas *fuzzy* criam uma interface entre padrões quantitativos e estruturas de conhecimento qualitativas, expressas em linguagem natural. Esta característica faz com que sistemas *fuzzy* sejam atraentes do ponto de vista da representação do conhecimento, permitindo que o conhecimento adquirido a partir de um banco de dados possa ser representado de forma compreensível. Como resultado, ele dá ao modelo um estágio superior de interpretabilidade(Hüllermeier, 2005).

Este artigo descreve um sistema para a indução automática de modelos, aplicados a tarefa de classificação. O modelo utilizado se chama Árvores de Padrões Fuzzy (APF). A APF é um modelo hierárquico baseado em árvores, tendo como nós internos, operadores lógicos *fuzzy* e as folhas das árvores são compostas por uma combinação de termos *fuzzy* e atributos de entrada. Neste modelo, o classificador é obtido, criando uma árvore para cada classe. Cada árvore será uma "descrição logica da classe", permitindo a interpretação dos resultados.

A estratégia para o aprendizado das árvores, proposta em (Senge e Hüllermeier, 2011) chamada "Beam Search", possui algumas limitações: A primeira está relacionada à característica gananciosa do algoritmo de busca, sempre procurando o melhor candidato no atual estágio da construção da árvore. Esta característica pode estreitar o espaço de busca, fazendo com que o algoritmo fique preso a uma solução sub ótima. Outra desvantagem é associada com a "Maldição da dimensionalidade". Se a quantidade de atributos e a largura do feixe forem grandes, o algoritmo levará muito tempo para avaliar todas as possibilidades, por esta razão, haverá uma explosão na quantidade de combinações possíveis. Por outro lado, se a largura do feixe é pequena, apenas uma pequena região do espaço de busca será explorada. Consequentemente limitando o algoritmo na tarefa de encontrar boas soluções. Para minimizar estes problemas, está sendo proposta a substituição dos métodos de aprendizado utilizados anteriormente, pela Programação Genética Cartesiana(PGC), pois a PGC é um méto-

do de busca global muito eficiente e que pode descobrir APF precisas e interpretáveis.

Embora, os métodos de indução de APF utilizados correntemente produzam resultados competitivos, propomos as seguintes mudanças: A primeira é a substituição do algoritmo de busca geralmente utilizado, pela Programação Genética Cartesiana (PGC). Baseando-se em que a PGC é uma forma de programação genética, muito eficiente e flexível, que codifica um programa de computador em uma representação em grafos. Assim, esse grafos são utilizados para representar várias estruturas computacionais, e podem ser facilmente usados para representar APFs. Além disso, a PGC é um método de pesquisa global capaz de explorar grandes espaços de busca eficientemente, buscando encontrar uma representação adequada de uma APF. A classificação feita através das árvores obtidas pela PGC devem apresentar uma boa acurácia , com árvores interpretáveis. A segunda alteração proposta é a possibilidade de empregar operadores *fuzzy* de concentração e de diluição. Estes operadores podem melhorar a representação linguística fornecida pela APF.

Este artigo é dividido em seis seções. Na segunda seção será apresentada uma descrição resumida das Árvores de Padrões Fuzzy. Na terceira será apresentada uma descrição concisa da PGC. Na quarta é descrito o método proposto e na quinta os resultados obtidos. Na sexta seção foram apresentadas as conclusões do trabalho.

2 Árvores de Padrões Fuzzy

As Árvores de Padrões Fuzzy foram criadas focando a representação do conhecimento através de uma expressão em forma de árvore ao invés de representa-lo em forma de regras. O primeiro método de indução de APF foi criado por Huang, Gedeon e Nikravesh(Huang et al., 2008), e mais tarde, o algoritmo de geração das árvores foi refinado em(Senge e Hüllermeier, 2011). A adoção deste tipo de representação hierárquica busca minimizar problemas existentes em sistemas baseados em regras, como o aumento exponencial da quantidade de regras com o aumento das entradas e a perda da interpretabilidade quando uma grande quantidade de regras for necessária para alcançar os requerimentos de acurácia. Como a árvore é representada como um grafo, ela favorece a habilidade humana de reconhecer padrões visuais, permitindo a descoberta de relações entre os atributos de entrada. Esta relação é difícil de ser feita quando são utilizados modelos com um conjunto fixo de regras. Consequentemente, APF fornecem uma alternativa para a construção de modelos *fuzzy* precisos e interpretáveis.

APF é um modelo hierárquico com uma estrutura de árvore, no qual os nós internos são operadores usados em sistemas *fuzzy* e as folhas são compostas por termos *fuzzy* associados com um atributo de entrada. No decurso da sua avaliação, a APF propaga a informação do fundo para o topo da árvore. Assim, os nós internos recebem valores de seus predecessores e os combina usando um operador. A saída da operação é apresentada ao nível superior da árvore. Consequentemente, a APF realiza um mapeamento recursivo, gerando a saída em um intervalo unitário. Um classificador baseado em árvores de padrões é desenvolvido pela criação de uma árvore para cada classe. A decisão do classificador ocorre em favor da árvore (classe) que possui o maior valor de saída. Também, desde que cada árvore é considerada uma "descrição lógica" da classe, ela permite uma interpretação mais especifica do problema de aprendizado(Senge e Hüllermeier, 2011).

Um exemplo de APF pode ser visto na Figura 1, a árvore representa a classe "Vinho de boa qualidade". Os atributos de entrada são o teor alcoólico, a acidez, a concentração de dióxido de enxofre e sulfatos. Eles são relacionados a termos *fuzzy* que representam um intervalo do universo de discurso de um atributo. Por exemplo, na Figura 1, o termo *fuzzy* Alcool_baixo, representa o conjunto *fuzzy* que indica um baixo teor alcoólico. Os valores de pertinência obtido nos conjuntos *fuzzy* são combinados por operadores que mantem os resultados parciais dentro do intervalo [0,1]. O valor obtido na saída após todas as combinações de características, deve se aproximar de 1, caso as características que compõe a árvore sejam úteis para a definição da classe que ela representa.

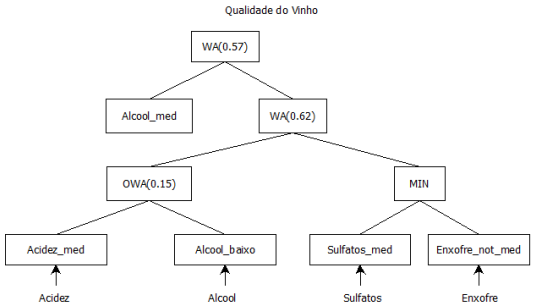

Figura 1. Árvore de Padrões Fuzzy de um vinho de boa qualidade.

A associação entre um atributo e um termo *fuzzy* é representada por uma função de pertinência. O modelo baseado em árvores mapeia diversas entradas em somente uma variável de saída. A interpretação da saída produzida pode ser vista como um modelo que simplifica a avaliação global de uma propriedade, avaliando diferentes critérios e agregando-os posteriormente(Senge e Hüllermeier, 2011). Observando a Figura 1, verifica-se que a qualidade do vinho está relacionada com duas sub-árvores. A primeira relaciona o teor alcoólico e a acidez, enquanto a outra relaciona a concentração de sulfatos e de dióxido de enxofre. As informações destas sub-árvores são posteriormente combinadas em um nível mais alto. Neste tipo de sistema é possível identificar que as sub-árvores representam diferentes conhecimentos que devem ser combinados. Também é relevante notar que a análise dessas sub-árvores pode fornecer informações suplementares. Dependendo da configuração da árvore, se um atributo está mais próximo do topo ou do fundo, pode-se identificar quais atributos contribuem mais para o resultado final. Além dos termos *fuzzy*, t-norms, t-conorms e operadores de média foram usados na criação das árvores(Senge e Hüllermeier, 2011).

3 Programação Genética Cartesiana

A PGC (Miller, 2011) é uma variação de programação genética na qual os programas são representados por grafos. O grafo é codificado em uma sequência linear de inteiros e são representados em uma grade de nós computacionais. Embora, a

grade possa ter qualquer número de dimensões, na maior parte das aplicações, ela apresenta somente uma ou duas. A *PGC* apresenta várias vantagens, tais como: Neutralidade, redundância e a ausência de um problema chamado "Bloat" (crescimento do programa sem retorno significativo em termos de aptidão), comum em outros métodos de programação genética (Miller, 2001; Miller e Smith, 2006). Na PGC, os grafos são representados em uma sequência linear de inteiros, que são chamados de cromossomos ou genótipos, como mostrado na Figura 2. Esses cromossomos podem ser divididos em partes. Estas partes são chamadas de genes. Os genes podem ser rotulados como gene de função, conexão ou de saída. Um nó é formado por um gene de função e dois genes de conexão. A função (Rotulada pelo gene de função) usa como parâmetros de entrada os valores indicados pelos genes de conexão para gerar a saída do nó, que por sua vez pode ser utilizada como parâmetro de entrada de uma função em outro nó. Por uma questão de desenvolvimento do processo evolutivo nos genótipos, é necessário decodificá-los em fenótipos para se obter a solução no domínio do problema. Quando o genótipo é decodificado, alguns nós podem não estar conectados ao fluxo de dados, criando um efeito de neutralidade(Miller, 2011). Um exemplo pode ser visto na Figura 2. Enquanto o genótipo tem um tamanho fixo, os fenótipos podem variar. O operador mutação é utilizado na PGC, sendo frequentemente utilizado o "one-point mutation".

A quantidade de genes que sofrem mutação é definida por uma porcentagem do total de genes, chamada de taxa de mutação. Nas Figuras 2 e 3 são mostrados o antes e depois de uma operação de mutação. Neste exemplo o último gene foi alterado; Pode-se ver que a mudança em um único gene pode transformar significativamente o fenótipo (Miller, 2011). O algoritmo evolucionário usado na PGC é a estratégia evolucionária 1+λ, onde o valor de λ frequentemente usado é quatro (Miller, 2011). Essa estratégia permite a PGC encontrar boas soluções de forma eficiente e em poucas avaliações. A próxima sessão descreve como a PGC pode ser usada para sintetizar APF.

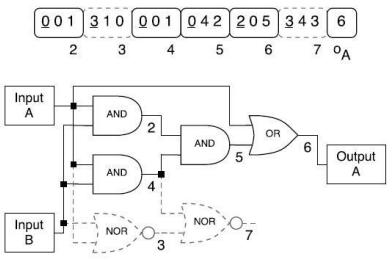

Figura 2. Genótipo (topo) e o respectivo fenótipo.

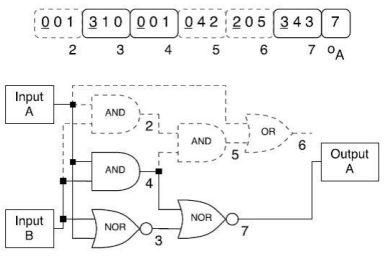

Figura 3. Genótipo-Fenótipo após mutação (o ultimo gene da sequencia sofreu mutação).

4 Método Proposto

O método proposto emprega a PGC na síntese de APF. Existem duas razões principais para justificar esta escolha. A primeira é a flexibilidade da representação em forma de grafos na PGC. Esta representação pode facilmente ser utilizada para expressar APF. A segunda razão é a competência da PGC em explorar grandes espaços de buscas de maneira eficiente, fazendo com que o processo de síntese seja menos sensível a maldição da dimensionalidade. Uma vez que a exploração do espaço de busca na PGC não depende da estratégia gananciosa do *Beam Search*, ela tem uma chance maior de conseguir soluções melhores. Além disso, não se restringe, como o *Beam Search* (que é limitado pela largura do feixe). Isso também aumenta as chances de obtenção de soluções melhores. O modelo proposto utiliza um algoritmo evolutivo, portanto seu êxito depende de como a solução é codificada e da escolha da função de aptidão para a avaliação da solução. As próximas duas subseções explicam cada uma dessas implementações.

4.1 Representação

Cada Atributo é rotulado por um de cinco termos linguísticos (termos *fuzzy*), os termos são, baixo, médio-baixo, médio, médio-alto e alto. Os termos linguísticos são obtidos pela partição uniforme do espaço dos atributos de entrada. Os genes de função podem representar os diferentes operadores que podem ser utilizados nas APF. No interesse de aumentar a facilidade de interpretação da solução (árvore), optou-se por utilizar um subconjunto dos operadores encontrados em (Senge e Hüllermeier, 2011). Foram utilizados os seguintes operadores:

$$Máximo = Max(a,b) \quad (1)$$

$$\text{Mínimo} = \text{Min}(a,b) \quad (2)$$

$$WA(k) = k \times a + (1-k) \times b \quad (3)$$

$$OWA(k) = k \times Max(a,b) + (1-k) \times Min(a,b) \quad (4)$$

Onde a e b são os valores das entradas dos nós que serão operados. No caso dos operadores WA e OWA, k será um valor criado de forma aleatória dentro do intervalo [0,1].

Modificadores e complemento também foram implementados, uma vez que podem fornecer um significado linguístico adicional. Foram adicionados dois inteiros associados aos genes de função para indicar que modificador ou complemento deve ser aplicado a cada entrada do operador. O primeiro inteiro indica qual modificador deve ser usado, enquanto o segundo indica o complemento. Por exemplo, se o primeiro inteiro é 0, significa que nenhum modificador será aplicado as entradas. Da mesma forma, se o segundo inteiro for 0, significa que não serão utilizados complementos nas entradas. Os modificadores (concentração ou diluição) e complemento são calculados como descrito abaixo:

$$Concentração = [\mu_a(x)]^2 \quad (5)$$

$$Diluição = [\mu_a(x)]^{\frac{1}{2}} \quad (6)$$

$$Complemento = 1 - [\mu_a(x)] \quad (7)$$

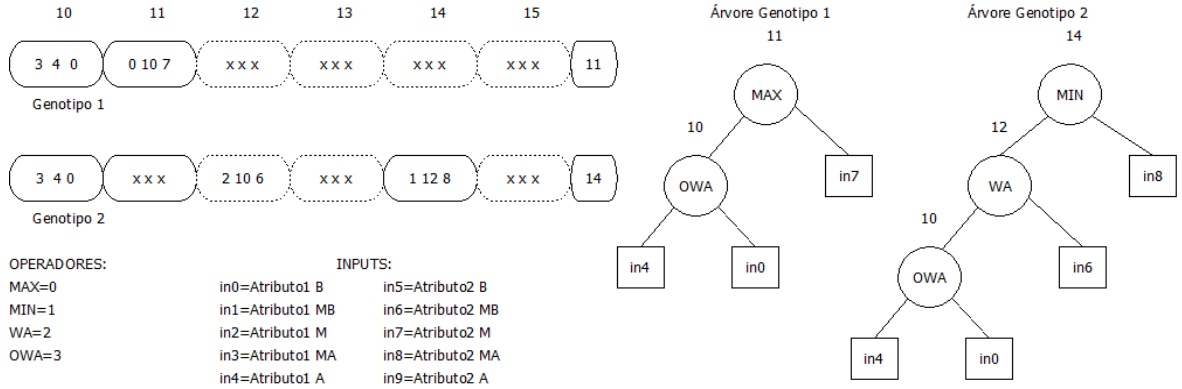

Figura 4. Representação de um par de genótipos e suas respectivas árvores.

Na Figura 4, um exemplo foi apresentado, com os valores de pertinência após cada modificação.

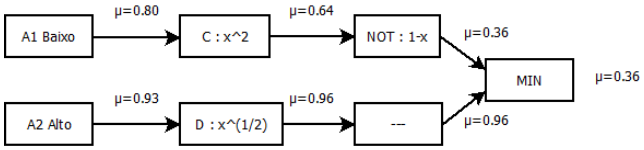

Figura 5. Exemplo de uma operação com o calculo do valor de pertinência.

A fim de obter o genótipo que irá representar a árvore, a PGC tem ao seu dispor todos os operadores e termos *fuzzy*. Dependendo do gene conexão utilizado como entrada do gene de função, diferentes árvores podem ser obtidas. A Figura 5 mostra dois genótipos diferentes e suas árvores equivalentes, para uma base de dados genérica com dois atributos e duas classes. O primeiro nó do genótipo 1 é (3,4,0). Isto significa que o operador OWA (3) tem as entradas in4(4) e in0(0). O segundo nó é (0,10,7), utiliza o operador Max(0) e possui como entradas in7(7) e a saída do primeiro nó rotulada pelo ponto de conexão 10. O gene de saída (11) mostra que a saída do segundo nó (ponto de conexão 11) deve ser utilizada como a saída da árvore. A árvore para o genótipo 2 pode ser construída de forma similar.

4.2 Função de aptidão

A função de aptidão utilizada é composta por duas partes e pode ser vista na equação 8. A primeira parte trata da finalidade de obter uma boa precisão e é calculada utilizando a medida RMSE.

$$Aptidão = w_1 \times AP_1 + w_2 \times AP_2 \tag{8}$$

$$RMSE = \frac{1}{N}\sqrt{\sum_{i-1}^{N}(o_i - d_i)^2} \tag{9}$$

$$AP_1 = Similaridade = 1 - RMSE \tag{10}$$

O erro é definido como a diferença entre o valor obtido na saída da árvore (o_i) e o valor alvo (d_i), que deve ser 1 se os valores apresentados nas entradas pertencem a classe representada pela árvore e 0 caso contrário. Após o cálculo do RMSE, o resultado é subtraído de 1 , resultando em uma medida de performance chamada de similaridade. A segunda parte da avaliação dos genótipos é descrita pela equação 11 e foi criada com o alvo de obter árvores menores e consequentemente, mais interpretáveis.

$$AP_2 = \frac{Qntdgenes - Genes\ Ativos}{Qntdgenes} \tag{11}$$

Como pode ser observado, o valor de AP_2 aumenta com a diminuição da quantidade de genes ativos. Consequentemente, árvores menores possuem uma melhor avaliação em AP_2. Portanto, guiando a busca da PGC para árvores menores, que são mais fáceis de explicar(interpretar) que árvores grandes.

5 Resultados

Esta sessão apresenta os estudos experimentais realizados para avaliar o desempenho do modelo proposto em diversas bases de dados. Estas bases foram obtidas no UCI *machine learning repository*. Elas também são usadas em(Senge e Hüllermeier, 2011), portanto, é possível comparar as APF criadas pelo método proposto com as criadas em(Senge e Hüllermeier, 2011). Adicionalmente, esses estudos fornecem uma comparação com outros classificadores bem conhecidos: Support Vector Machine com Kernel Linear (SVM-L)(Vapnik, 1999), K-nearest neighbors(KNN)(Webb, 2002), Random Forest (RF) (Breiman, 2001) e Support Vector Machine com Kernel Radial Basis Function(SVM-R)(Vapnik, 1999).

O critério confrontado foi a estimativa da medida de desempenho de generalização realizada através de uma validação cruzada com 10 pastas e 5 repetições. Duas medidas de desempenho foram utilizadas: a acurácia da classificação e a área sob a curva ROC (AUC). A AUC fornece uma medida mais robusta que a acurácia para avaliar modelos de classificação. Infelizmente, ela só pode ser usada em problemas de classificação binários, portanto, quando a base de dados possui mais que duas classes, a AUC é calculada pela média das AUC estimadas no modo “Um contra todos”. Os parâmetros dos outros classificadores foram definidos da seguinte forma: O número de vizinhos do KNN foi fixado em 1; a quantidade de

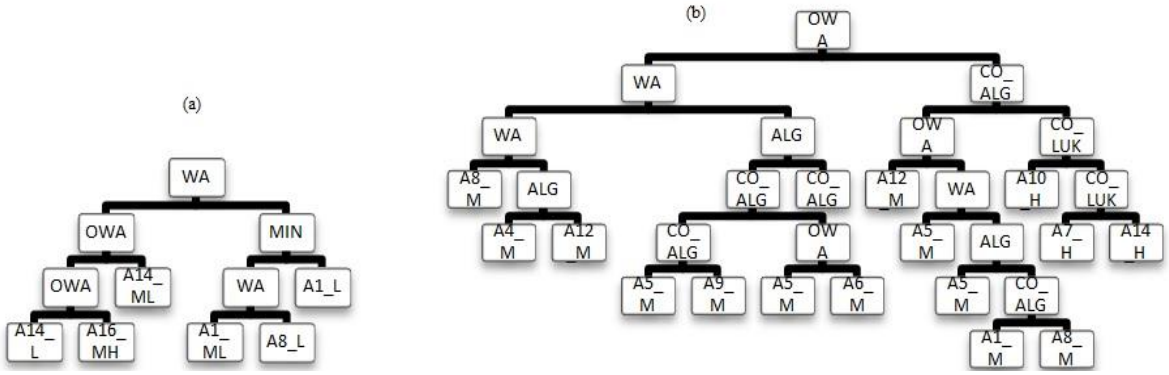

Figura 6. (a) Base de dados Australian, classe 1 APF-PGC2. (b) Base de dados Australian, classe 1 PTTDE.25

árvores geradas no Random Forest foi fixada em 50 e a quantidade de atributos sob o qual é feita a partição é igual a 1.

Os parâmetros do SVM foram determinados utilizando validação cruzada. Uma vez que os parâmetros ótimos foram encontrados, o classificador foi treinado com o conjunto de treinamento e seu desempenho foi avaliado no conjunto de teste. O procedimento usado para encontrar os parâmetros não foi tendencioso, pois a pasta de teste na validação não foi utilizada para ajustar os parâmetros. Apesar da validação cruzada requerer um maior custo computacional, ela fornece uma medida da habilidade de generalização, pois as pastas de teste não foram utilizadas para ajustar os parâmetros; Além disso, também ajuda a reduzir o *overfit*.

Duas Variantes das APF-PGC sem modificador/complemento foram criadas: APF-PGC1 que possui genótipos com 400 nós evoluídos em 600 gerações. O APF-PGC2 possui genótipos com 50 nós evoluídos em 1250 gerações. Uma terceira variante APF-PGC3 possuindo genótipos com 50 nós evoluídos em 1250 gerações foi implementado utilizando os *fuzzy hedges*(Modificadores e complemento). Nessas três variantes, w1=0.7 e w2=0.3. A tabela 1 apresenta a acurácia e a tabela 2 apresenta a AUC. Nessas tabelas, as medidas de desempenho foram arredondadas para duas casas decimais. A análise das tabelas mostra que o APF-PGC2 apresenta resultados competitivos. Em relação a acurácia, ele apresentou o melhor resultado em 3 bases de dados, acurácia igual ou maior que 90% em 5 bases de dados e uma diferença absoluta da acurácia para o melhor algoritmo de menos que 5% em 11 bases de dados. Quando o AUC é comparado, pode-se ver que foi obtido o melhor resultado em 5 bases de dados, AUC igual ou maior que 0.9 em 5 bases de dados e diferença absoluta do melhor resultados menor ou igual a 5% em 10 bases de dados. A tabela 1 também apresenta o PTTDE.25, que é o modelo com o melhor resultado de acurácia nas implementações anteriores de APF(Senge e Hüllermeier, 2011). Esse resultados foram obtidos da implementação em java do algoritmo "Top-Down" fornecido pelos autores(Senge e Hüllermeier, 2011). Neste caso, somente os resultados referentes a acurácia estão disponíveis.

Tabela 1 Resultados-Acurácia

Datasets	SVM-L	KNN	RF	SVM-R	APF-PGC1	APF-PGC2	PTTDE.25*	APF-PGC3
Iris	**0.96**	0.94	0.95	0.95	**0.96**	**0.96**	0.95	0.92
Wine	**0.98**	0.95	**0.98**	**0.98**	0.91	0.93	**0.98**	0.90
Sonar	0.80	0.87	0.88	**0.91**	0.77	0.77	0.80	0.71
Pima	**0.77**	0.70	**0.77**	0.71	0.73	0.75	0.76	0.73
Balance	0.87	0.77	0.85	**0.95**	0.83	0.84	0.87	0.75
Habeman	0.72	0.67	0.65	0.71	0.75	**0.76**	0.74	0.71
Breast_Cancer	**0.97**	0.95	0.96	0.96	0.96	0.96	0.96	0.96
Australian	**0.86**	0.80	0.79	0.85	0.85	0.84	0.85	0.85
Ionosphere	0.87	0.86	0.90	**0.92**	0.88	0.90	0.91	0.83
Lupus	0.74	0.68	0.62	0.73	0.73	0.75	**0.77**	**0.77**
Bupa	0.67	0.61	0.66	0.67	0.68	0.65	**0.70**	0.63
Transfusion	0.76	0.71	0.70	0.73	**0.77**	**0.77**	**0.77**	0.76
Lawsuit	**0.99**	0.96	0.97	0.96	0.96	0.96	0.94	0.97

Tabela 2 Resultados-AUC

Datasets	SVM-L	KNN	RF	SVM-R	APF-PGC1	APF-PGC2	APF-PGC3
Iris	0.93	0.98	0.99	0.99	0.99	**1.00**	**1.00**
Wine	**1.00**	0.98	**1.00**	**1.00**	0.99	0.99	0.96
Sonar	0.89	0.95	0.94	**0.98**	0.85	0.84	0.76
Pima	**0.83**	0.72	**0.82**	0.79	0.79	0.77	0.78
Balance	0.93	0.88	0.95	**1.00**	0.91	0.94	0.85
Habeman	0.70	0.62	0.57	0.63	0.67	**0.75**	0.68
Breast_Cancer	**0.99**	**0.99**	**0.99**	0.98	**0.99**	**0.99**	0.98
Australian	**0.93**	0.86	0.92	0.91	0.91	0.89	0.90
Ionosphere	0.87	0.97	0.97	**0.98**	0.92	0.90	0.81
Lupus	0.86	0.67	0.72	0.78	0.84	**0.89**	0.82
Bupa	0.71	0.64	0.72	**0.73**	0.68	**0.73**	0.67
Transfusion	**0.74**	0.60	0.64	0.66	0.72	0.70	0.66
Lawsuit	**1.00**	0.97	0.99	0.99	0.99	0.99	0.96

Tabela 3 Profundidade média das árvores

Datasets	APF-PGC2	PTTDE.25	APF-PGC3
Iris	2.60	3.33	**1.86**
Wine	2.33	10.0	**1.36**
Sonar	1.50	10.0	**1.10**
Pima	2.00	4.00	**1.60**
Balance	4.30	**1.66**	4.00
Habeman	6.00	2.00	**2.00**
Breast_Cancer	2.75	5.00	**2.5**
Australian	2.00	5.00	**1.90**
Ionosphere	2.00	14.0	**1.35**
Lupus	4.75	4.00	**3.40**
Bupa	4.25	10.0	**2.85**
Transfusion	4.75	5.00	**2.60**
Lawsuit	4.50	9.00	**2.05**

Para confirmar estes achados, foram aplicados os testes de Friedman e o Nemenyi Post-Hoc. Eles foram utilizados para comparar o desempenho dos classificadores e para verificar se é possível identificar um classificador que apresente um desempenho melhor, que seja estatisticamente significante (nível de significância dado por p-value=0.05) quando se leva em consideração todas as diferentes bases de dados dos experimentos. Com relação às medidas de acurácia e AUC, os testes revelaram que não há diferença estatisticamente significativa entre os classificadores. A segunda comparação feita trata da profundidade média das árvores do APF-PGC2, APF-PGC3 e PTTDE.25. Esta comparação foi feita para o pior caso de APF-PGCs, onde o peso da função de aptidão que favorece

árvores menores foi fixado em zero, w_1=1 e w_2=0. Dos resultados apresentados na tabela 3, que também foram confirmados pelo teste de Friedman, pode-se concluir que os APF-PGC2 e APF-PGC3 tiveram um melhor desempenho que o PTTDE.25 com respeito a profundidade média das árvores. Os resultados indicam que o método de busca "Beam-search" utilizado no PTTDE.25, tem um desempenho inferior, devido a sua característica gananciosa, pois ele escolhe as melhores sub-árvores candidatas no momento, sem voltar atrás, descartando candidatos que podem não ter um bom desempenho individualmente, mas quando combinados com outra sub-árvore escolhida posteriormente poderiam ter um desempenho melhor. A Figura 6 apresenta exemplos de árvores criadas pelos APF-PGC2 e pelo PTTDE.25. A tabela 3 mostra também que o uso de modificadores contribui para o desenvolvimento de árvores menores. As Figuras 7 e 8 apresentam uma comparação entre as árvores criadas pelo APF-PGC2 e APF-PGC3 para um problema de classificação de vinhos. Neste problema deseja-se classificar vinhos de três vinícolas, utilizando uma base de dados contendo treze atributos. A árvore da Figura 7 apresenta o grau de pertinência do vinho em relação a vinícola 1. Desde que o APF-PGC3 utiliza modificadores e complementos, é possível criar uma expressão mais compacta que o APF-PGC2. A expressão obtida pela árvore é:

A13 é um pouco Médio-Alto
E
A4 é Não-Alto e A12 é um pouco Médio-Alto

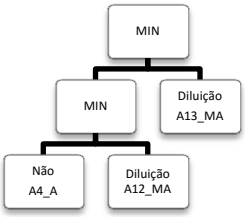

Figura 7. Base de dados Wine, classe 1. APF-PGC3

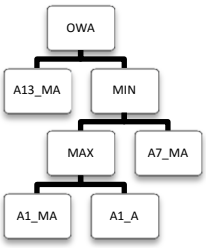

Figura 8. Base de dados Wine, classe 1. APF-PGC2

O APF-PGC3 utiliza somente três atributos para fazer a classificação. Esta é outra vantagem da utilização de APF, seu algoritmo de indução apresenta uma seleção de atributos. O atributo 4(A4) representa a alcalinidade das cinzas presentes no vinho, o A12 representa OD280\D350 de vinho diluídos e A13 indica a concentração de proline, um aminoácido presente nas uvas. Pode ser visto que a concentração de proline(A13) é mais importante que os outros atributos, pois ela ocupa uma posição mais elevada na árvore gerada(mais perto da saída). Adicionalmente, pode ser visto que isto também ocorre na árvore gerada pelo APF-PGC2, indicando a importância de uma concentração Média-Alta de proline para a classificação de vinhos da vinícola 1.

6 Conclusão

Este artigo propõe uma nova forma de indução de Árvores de Padrões Fuzzy utilizando Programação Genética Cartesiana como algoritmo de aprendizado. Os resultados obtidos demostraram que APF-PGC tem um desempenho competitivo na tarefa de classificação em relação a alguns dos melhores classificadores disponíveis, com a vantagem de fornecer um modelo interpretável, ou seja, o conhecimento obtido no aprendizado pode ser extraído do modelo. APF constituem uma alternativa viável aos clássicos modelos *fuzzy* baseados em regras, pois sua estrutura hierárquica permite uma representação mais compacta e um compromisso entre a acurácia e a simplicidade do modelo.

Referências Bibliográficas

Breiman, L., 2001. Random Forests. Mach. Learn. 45, 5–32.

Cordón, O., 2011. A Historical Review of Evolutionary Learning Methods for Mamdani-type Fuzzy Rule-based Systems: Designing Interpretable Genetic Fuzzy Systems. Int J Approx Reason. 52, 894–913.

Herrera, F., 2008. Genetic fuzzy systems: taxonomy, current research trends and prospects. Evol. Intell. 1, 27–46.

Huang, Z., Gedeon, T.D., Nikravesh, M., 2008. Pattern Trees Induction: A New Machine Learning Method. IEEE Trans. Fuzzy Syst. 16, 958–970.

Hüllermeier, E., 2005. Fuzzy Methods in Machine Learning and Data Mining: Status and Prospects. Fuzzy Sets Syst 156, 387–406.

Miller, J., 2001. What bloat? Cartesian Genetic Programming on Boolean problems. pp. 295–302.

Miller, J.F., 2011. Cartesian Genetic Programming, 2011 edition. ed. Springer, Heidelberg : New York.

Miller, J.F., Smith, S.L., 2006. Redundancy and computational efficiency in Cartesian genetic programming. IEEE Trans. Evol. Comput. 10, 167–174.

Senge, R., Hüllermeier, E., 2011. Top-Down Induction of Fuzzy Pattern Trees. Trans Fuz Sys 19, 241–252.

Vapnik, V., 1999. The Nature of Statistical Learning Theory, 2nd edition. ed. Springer, New York.

Webb, A.R., 2002. Density Estimation – Nonparametric, in: Statistical Pattern Recognition. John Wiley & Sons, Ltd, pp. 81–122.

Witten, I.H., Frank, E., 2005. Data Mining: Practical Machine Learning Tools and Techniques, Second Edition. Morgan Kaufmann Publishers Inc.