Universidade do Estado do Rio de Janeiro

Centro de Tecnologia e Ciências

Escola de Engenharia

Cesar Eduardo Rodrigues Ferreira de Almeida

APFR-PGC: Síntese de Árvores Padrões *Fuzzy* de Regressão via Programação Genética Cartesiana

Rio de Janeiro

2017

Cesar Eduardo Rodrigues Ferreira de Almeida

APFR-PGC: Síntese de Árvores Padrões Fuzzy de Regressão via Programação Genética Cartesiana

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao programa de Pós-Graduação em Engenharia Eletrônica da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes e Automação.

Orientador: Prof. Dr. Jorge Luís Machado do Amaral

Rio de Janeiro

2017

CATALOGAÇÃO NA FONTE

UERJ / REDE SIRIUS / BIBLIOTECA CTC/B

A447 Almeida, Cesar Eduardo Rodrigues Ferreira de.
APFR-PGC: síntese de árvores padrões fuzzy de regressão via programação genética cartesiana / Cesar Eduardo Rodrigues Ferreira de Almeida. – 2017.
80f.

Orientador: Jorge Luís Machado do Amaral.
Dissertação (Mestrado) – Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia.

1. Engenharia Eletrônica - Teses. 2. Lógica difusa - Teses. 3. Programação genética (Computação) - Teses. 4. Algoritmos - Teses. I. Amaral, Jorge Luís Machado do. II. Universidade do Estado do Rio de Janeiro. III. Título.

CDU 681.52

Assinatura

Data

Cesar Eduardo Rodrigues Ferreira de Almeida

APFR-CGP: Síntese de Árvores Padrões Fuzzy de Regressão via Programação Genética Cartesiana

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao programa de Pós-Graduação em Engenharia Eletrônica da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes e Automação.

Aprovada em:

Banca Examinadora:

_____

Prof. Dr. Jorge Luís Machado do Amaral D.Sc (Orientador)

Faculdade de Engenharia – UERJ

_____

Prof. Dr. Douglas Mota Dias D.Sc

Faculdade de Engenharia – UERJ

_____

Prof. Dr. Ricardo Tanscheit D.Sc

Departamento de Engenharia Elétrica – PUC-Rio

Rio de Janeiro

2017

DEDICATÓRIA

Dedico este trabalho a minha família, Eva, Kácia e Almeida que me deram apoio para esta etapa da minha vida.

AGRADECIMENTOS

Ao meu orientador Professor Jorge Amaral pelo estímulo, parceria e paciência para a realização desta dissertação.

À UERJ e ao Programa de Pós-Graduação em Engenharia Eletrônica, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

À minha família, pela paciência e carinho.

Aos meus pais, pela educação, atenção e carinho de todas as horas.

Aos meus amigos por todo apoio e compreensão.

Aos professores e funcionários do Programa de Pós-Graduação em Engenharia Eletrônica e do Departamento de Engenharia Eletrônica e de Telecomunicações.

Aos meus amigos Anderson e Carlos que de uma forma ou de outra me estimularam ou me ajudaram.

Há homens que lutam um dia e são bons, há outros que lutam um ano e são melhores, há os que lutam muitos anos e são muito bons. Mas há os que lutam toda a vida, estes são imprescindíveis.

Bertolt Brecht

RESUMO

ALMEIDA, Cesar Eduardo Ferreira de. *APFR-CGP: síntese de árvores padrões fuzzy para problemas de regressão via programação genética cartesiana*. 2017. 80f. Dissertação (Mestrado em Engenharia Eletrônica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2017.

Este trabalho explora um modelo alternativo que emprega a teoria dos conjuntos fuzzy para resolver problemas de regressão. Ao invés do tradicional Sistema Fuzzy Baseado em Regras foi utilizada a estrutura hierárquica denominada Fuzzy Pattern Trees for Regression (Árvores de Padrões Fuzzy para regressão-APF), que representa o conhecimento de forma mais compacta e fornece um compromisso entre acurácia e interpretabilidade. Esta estrutura na forma de árvore é composta por folhas que são termos fuzzy associados aos atributos, e de nós que são os operadores utilizados em sistemas fuzzy. Já as saídas são aproximações dos valores reais de funções que foram sintetizadas para resolver o problema de regressão. O algoritmo para síntese foi substituído pela Programação Genética Cartesiana (PGC), que consegue explorar grandes espaços de busca de forma eficiente. Neste trabalho foram criados dois modelos que exploram a sinergia das APFs e da PGC. A parte experimental realizada através de base de dados (BD) disponíveis no UCI e KEEL buscou achar uma melhor configuração geral para as árvores e comparar os modelos aqui criados com os métodos k-vizinhos mais próximos, Regressão Linear, Árvores de Regressão, Máquinas de Vetores de Suporte e *Multilayer Perceptron*. Além disso, houve a comparação com o método original das APFs. Os resultados obtidos se mostraram competitivos em termos de acurácia e de interpretabilidade.

Palavra-chave: Árvores de Padrões Fuzzy; Programação Genética Cartesiana; Regressão; Interpretabilidade.

ABSTRACT

ALMEIDA, Cesar Eduardo Ferreira de. *FTP-CGP: s*ynthesis of fuzzy pattern trees for regression by cartesian genetic programming. 2017. 80f. Dissertação (Mestrado em Engenharia Eletrônica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2017.

This work explores an alternative model uses fuzzy set theory to solve regression problems. Instead of the traditional fuzzy rule based system, a hierarchical structure called Fuzzy Pattern Trees for Regression, which represents knowledge in a more compact way and presents a compromise between security and interpretability, has been used. This structure in the form of a tree is composed of leaves that are fuzzy terms associated with the attributes. The outputs are approximations of the real values of functions that are synthesized to solve the regression problem. The algorithm for synthesis was replaced by Cartesian Genetic Programming, which can efficiently explore large search spaces. In this work two models were created that exploit a synergy of APFs and PGC. The experimental part performed through available data sets in the UCI and KEEL repositories sought to find a better overall configuration for the trees and compare models created here with the k nearest neighbors, Linear Regression, Regression Trees, Support Vector Machines and Multilayer Perceptrons. In addition, a comparison was made with the original method of APFs. Results were competitive in terms of accuracy and interpretability.

Keywords: Fuzzy Pattern Trees; Cartesian Genetic Programming; Regression; Interpretability.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

INTRODUÇÃO

Atualmente, o volume de dados armazenado continua crescendo rapidamente. Infelizmente, devido à dificuldade do ser humano de interpretar tamanha quantidade de dados, muita informação e conhecimento podem estar sendo desperdiçados, ficando ocultos dentro das Bases de Dados. Neste contexto, surgiu a área multidisciplinar chamada “mineração de dados” (MD), que, de acordo com (MONARD; BARANAUSKAS, 2003), incorpora técnicas utilizadas em diversas áreas como inteligência computacional (IC) e Aprendizado de Máquinas. A MD se utiliza de diferentes algoritmos para explorar grandes quantidades de dados à procura de padrões e relacionamentos ocultos, com o objetivo de auxiliar uma tomada de decisão. De modo geral a indução dos modelos é feita de forma automática por diversas abordagens, dentre as quais se podem citar: Redes Neurais Artificiais (YEGNANARAYANA, 2009), métodos bayesianos, modelos gráficos, árvores de decisão, entre outros. (WITTEN; FRANK, 2005).

Um ponto negativo é que nem todas as técnicas que utilizam MD conseguem fornecer uma explicação de como o modelo obtém sua decisão, ou mesmo disponibilizar o conhecimento obtido na construção do modelo. Um paradigma bem atraente, que consegue traduzir em termos matemáticos informações imprecisas é a teoria dos conjuntos *fuzzy* (GOGUEN, 1973). O seu uso apresenta a vantagem da criação de uma interface entre padrões quantitativos e estruturas de conhecimentos qualitativos expressos em termos de linguagem natural. Assim, a representação do conhecimento é feita de uma forma atraente, pois a forma linguística que ela fornece é facilmente compreensível e interpretável (HÜLLERMEIER, 2005). Primordialmente, os sistemas *fuzzy* foram utilizados para aplicações na área de controle. Com o passar do tempo houve uma série de novas aplicações, e até hoje são considerados um paradigma importante na área de IC.

O uso mais frequente da teoria dos conjuntos *fuzzy* reside na criação de Sistemas *Fuzzy* Baseados em Regras (SFBR). Estes, através de um conjunto de regras “Se-Então” conseguem representar o conhecimento de um certo problema.

Os SFBR podem ser sintetizados com o auxílio de um especialista ou através de métodos numéricos (MENDEL, 2017). Um dos principais problemas que afetam a síntese dos SFBR é o que se chama de “maldição da dimensionalidade”. Isto ocorre quando um acréscimo linear de variáveis de entrada resulta em um aumento exponencial do número de parâmetros ajustáveis (USBERTI, 2016), o que torna a síntese de um SFBR, em muitas situações, uma tarefa difícil, pois, se o número de variáveis for grande, o número de possíveis regras cresce, dificultando a busca por um conjunto de regras adequado à solução do problema. Além disso, às vezes é necessário um grande número de regras para o sistema obter uma acurácia aceitável. Entretanto, um grande número de regras resulta em um maior esforço computacional e também em uma diminuição da interpretabilidade, ou seja, a capacidade de expressar o comportamento do sistema real de forma compreensível. (GACTO; ALCALÁ; HERRERA, 2011)

Com o intuito de obter melhores resultados, pesquisadores concatenaram diferentes paradigmas para formar sistemas híbridos. Estes têm como objetivo aproveitar os pontos fortes das técnicas tentando minimizar suas deficiências. Dentre os modelos híbridos, os sistemas *Fuzzy-Evolucionários* (SFE) tem grande importância para este trabalho, pois combinam o uso dos denominados Algoritmos Evolucionários (AE) com a teoria dos conjuntos *fuzzy*.

Os SFEs vêm sendo usados em diversos trabalhos, tais como: (HERRERA, 2008), (KARR, 1991), (THRIFT, 1991), (KOSHIYAMA; VELLASCO; TANSCHEIT, 2016) e (SANTOS; DO AMARAL, 2014). Alguns desses utilizam algoritmos genéticos (AG) em conjunto com sistemas fuzzy. Já em (SANTOS; DO AMARAL, 2014), utilizaram-se as Árvores de Padrões *Fuzzy* (APF) sintetizadas pela meta heurística genética (MHG) chamada Programação Genética Cartesiana (PGC), para resolver especificamente problemas de classificação.

As APFs são modelos hierárquicos que inicialmente foram criadas por (HUANG; GEDEON; NIKRAVESH, 2008) e depois aperfeiçoadas por (SENGE;HULLERMEIER, 2011). Utilizam uma estrutura na forma de árvore, onde suas folhas são termos fuzzy associados aos atributos e os nós são os operadores utilizados em sistemas *fuzzy*. Basicamente, esse algoritmo programa uma função que mapeia uma combinação de atributos de entrada em um número no intervalo

[0,1]. De acordo com (SENGE; HULLERMEIER, 2011), a APF é um modelo bem atrativo, pois no quesito interpretabilidade consegue-se perceber a lógica de tomada de decisão e qual atributo é mais relevante para a saída. Uma desvantagem do modelo original está relacionada ao método de aprendizado. Para sintetizar as APFs em (SENGE; HULLERMEIER, 2011) foi utilizado o método *Top-Down induction*, que utiliza a estratégia de busca denominada *Beam Search* (ZHANG, 1998). Esta pode achar soluções em sub ótimos globais, pois, a exploração do espaço de busca é feita de forma “gulosa”.

Uma derivação da Programação Genética (PG) que é importante para este trabalho é a Programação Genética Cartesiana (PGC) (MILLER, 2011). Esta representa programas escritos na forma de grafos, que são codificados em uma sequência linear de inteiros, representados por uma grade de nós computacionais de n-dimensões. Os genótipos representam a sequência linear de números inteiros, onde, cada número inteiro é chamado de gene, que pode ser rotulado de função, conexão ou saída. Cada nó é representado pela junção de genes, em que o gene de função utiliza os genes de conexão para gerar uma saída através de um conjunto de funções previamente definidas. A PGC é um método de busca global capaz de explorar grandes espaços de busca de forma eficiente, e pode representar os grafos utilizados nas APFs.

Neste trabalho, buscando explorar um método alternativo ao tradicional SFBR para problemas de regressão, criaram-se modelos híbridos do tipo *Fuzzy*-Evolucionário. Propõe-se, portanto, dois modelos que utilizam as denominadas APFs em conjunto com a MHG denominada PGC.

Objetivo

O propósito deste trabalho foi explorar um método alternativo aos SFBR que proporcione ao usuário uma estrutura de boa interpretação e acurácia. Para tal, criaram-se dois modelos híbridos denominados APFR-CGP para problemas de regressão.

O modelo 1 usa um genótipo com uma única saída, que representa valores entre [0,1]. Pode-se pensar como sendo um grau de qualquer função de pertinência fuzzy, cujo valor no mundo real pode ser encontrado através da inversa da função utilizada.

No modelo 2 utilizam-se diversos genótipos que representam múltiplas APFs. Cada uma dessas árvores representa uma função normalmente utilizada nos conjuntos fuzzy. Para se achar o valor real do problema é necessário um método de defuzzificação (ROSS, 2009), tal como altura, média dos máximos, centroide, etc... .

Esta dissertação também tem as seguintes finalidades:

- Encontrar a configuração mais adequada para as APFs que consigam de forma eficaz encontrar soluções satisfatórias em termos de acurácia e interpretabilidade. Para tal, analisam-se as entradas, saídas e operadores *fuzzy* mais adequados;
- Analisar o impacto de *Modificadores* (Intensificadores, diluidores e complemento);
- Fazer um comparativo com os métodos k-vizinhos mais próximos, Regressão Linear, Árvores de Regressão, Máquinas de Vetores de Suporte, *Multilayer Perceptrons*. Além disso, foi realizada a comparação com o método original das APFs.

Descrição e organização da dissertação

Este trabalho está organizado da seguinte forma:

- O capítulo 1 faz uma revisão bibliográfica do principal modelo utilizado em regressão importante para a execução deste trabalho. São também abordados os Sistemas *Fuzzy* com ênfase em Sistemas *fuzzy* evolucionários;
- O capítulo 2 descreve os conceitos básicos das Árvores de Padrões *Fuzzy*;

- O capítulo 3 apresenta de forma sucinta a Programação Genética Cartesiana;
- O capítulo 4 introduz o modelo hibrido APFR-CGP para problemas de regressão;
- O capítulo 5 contém a delineamento experimental para se encontrar a melhor configuração e o comparativo com métodos clássicos;
- Por último, apresenta-se a conclusão e os possíveis trabalhos futuros.

1 Sistemas fuzzy para regressão

Este capítulo tem o objetivo de fazer uma breve revisão bibliográfica de alguns modelos de sistemas fuzzy utilizados em regressão, tais como: sistemas *fuzzy* baseados em regras, sistemas n*euro fuzzy* e sistemas fuzzy evolucionários.

1.1 Sistemas fuzzy

Um paradigma bem atraente que consegue traduzir em termos matemáticos informações imprecisas ou vagas é a teoria dos conjuntos *fuzzy*. Os sistemas nebulosos, como também são chamados, tiveram o marco inicial no artigo *Fuzzy Sets* (ZADEH, 1965). Esta teoria tem um grande campo de aplicações que demonstra grande eficácia em problemas de controle (ZIMMERMANN, 1996), regressão (ASAI, 1982), previsão de séries temporais. (SONG; CHISSOM, 1993).

Um método que utiliza a teoria dos conjuntos fuzzy para aproximar a previsão de valores de função continua é o algoritmo de *Wang-Mendel* (WANG; MENDEL, 1992a). O objetivo do método é a geração de regras fuzzy a partir de dados numéricos. Este consegue aproximar valores de funções continuas em um conjunto compacto para precisão arbitrária (WANG; MENDEL, 1992b). No entanto, quando o número de variáveis é grande, o número de possíveis regras aumenta exponencialmente, assim criando a denominada “maldição da dimensionalidade”. Esta torna a busca do conjunto de regras difícil, aumentando o esforço computacional, o que pode ocasionar a diminuição de desempenho em tempo real e da interpretabilidade (TORRA, 2002).

O método de *Wang-Mendel* sugere cinco passos para a geração de regras (MENDEL, 1995).

1. Divisão do conjunto de dados de entradas e saídas em conjuntos *Fuzzy*;
2. Geração das regras *Fuzzy* a partir dos pares de dados de entrada e saída;
3. Designação de um grau para cada regra gerada;
4. Combinação da base de regras *Fuzzy;*
5. Procedimento de defuzzificação para a obtenção do mapeamento da entrada/saída da base de regras fuzzy combinadas.

1.1.1 Sistemas fuzzy baseados em regras

Os SFBR conseguem representar o conhecimento de certo problema e são estruturas conforme mostrado na Figura 1.

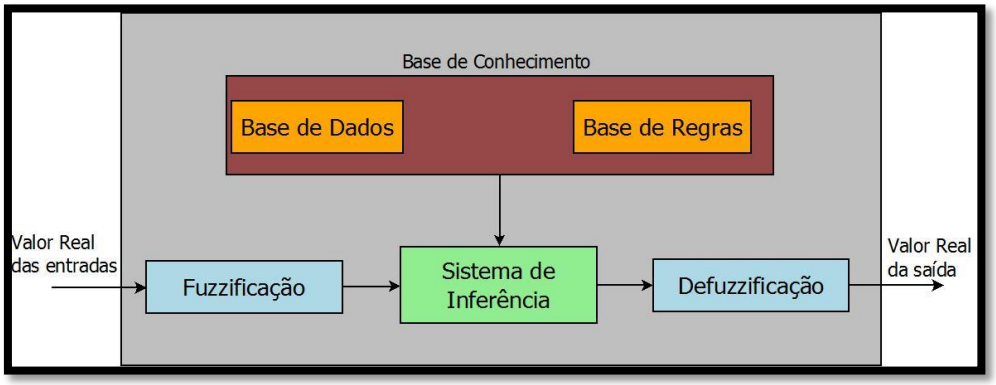

Figura 1-Modelo de um SFBR

Na fuzzificação ocorre o mapeamento da base de dados em conjuntos *fuzzy* através das funções de pertinência. Esta fase é responsável pela ativação das regras para determinadas situações (TANSCHEIT, 2004).

A base de conhecimento pode ser dividida em base de dados e base de regras. A primeira compreende os termos linguísticos que são utilizados nas regras, onde as funções de pertinência definem o rótulo linguístico e as funções de escala[1]. Além disso, cada variável linguística é associada a uma partição *fuzzy* criada na

[1] As funções de escala são utilizadas na transformação entre o universo de discurso onde os conjuntos fuzzy são definidos e o domínio de variáveis de entrada e saída. (HERRERA, 2008)

etapa de fuzzificação. A base de regras contém as sentenças linguísticas elaboradas por um especialista do problema ou extraída de dados. As regras são compostas por antecedentes e consequentes. Além disso, existem estruturas que podem influenciar na semântica das regras, dentre as quais podemos citar: conectivos lógicos (e, ou,..) e modificadores (intensificadores, diluidores, etc..). As bases de regras são geralmente definidas por um “Se-Então”. Um exemplo de uma regra está na Figura 2, onde, a cláusula entre o “se” e o “então” é denominada antecedente e a sentença após do “então” é o consequente (REZENDE, 2003). Esta regra ainda conta com o conectivo e com os *modificadores* (pouco e muito).

Na parte de inferência são determinadas as regras que serão ativadas e combinadas. Como resultado desse processo, é gerado o conjunto fuzzy de saída.

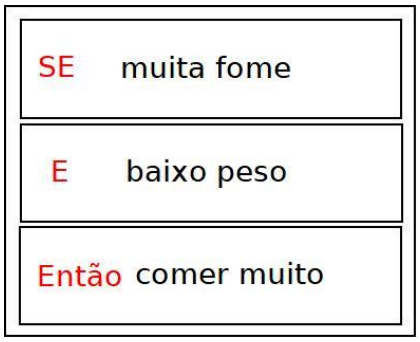

Figura 2-Exemplo de regra

Na defuzzificação ocorre a interpretação da informação contida no conjunto fuzzy de saída para valores *reais,* uma vez que, na maioria das aplicações, o que se deseja são valores no mundo real e não valores de grau de pertinência. Existem vários métodos de defuzzificação, tais como: altura, media dos máximos, centroide, etc... .(ROSS, 2009)

Os sistemas de inferência *fuzzy* propostos por *Zadeh* começaram a ter aplicabilidade prática por volta da década de 70. Dentre os sistemas de inferência mais conhecidos pode-se destacar o do tipo *Mamdani* (MAMDANI, 1974) e o do tipo *Takagi-Sugeno* (TAKAGI; SUGENO, 1983). No primeiro, o processo de decisão baseado em regras utiliza a estrutura “Se-Então”, onde tanto antecedente como consequente são valores de variáveis linguísticas, expressos por meio de conjuntos fuzzy (MAMDANI; ASSILIAN, 1975). Este sistema provou ser eficaz para aplicações que necessitam de interpretabilidade (TORRA, 2002). O sistema *Takagi-Sugeno* é

outro sistema de inferência que tem grande aceitação. Igualmente ao *Mamdani*, utiliza a relação entre antecedentes e consequentes da estrutura. Seu antecedente é formado por valores de variáveis linguísticas, já seu consequente é uma função linear dos seus antecedentes. Isto é, o formato de regra geral pode ser escrito como:

$$R_i: Se\ x \text{ é } A_i\ e\ y \text{ é } B_i\ então\ z = f_i(x, y)$$

Onde i varia de 1 até o número total de regras, x e y são as variáveis de entrada e z é a função de saída. Outro fato interessante é que o *Takagi-Sugeno* pode ser utilizado como um aproximador universal de qualquer sistema não linear (TAKAGI; SUGENO, 1985). Primordialmente, os sistemas fuzzy foram utilizados para aplicações na área de controle e com o passar do tempo houve uma série de novas aplicações. Pesquisadores na busca da sinergia entre modelos de inteligência computacional combinaram paradigmas criando os sistemas híbridos, como os sistemas *Neuro-Fuzzy* e o *Fuzzy-Evolucionário*. Em sistemas *Neuro-Fuzzy* utiliza-se o processamento linguístico de um sistema de inferência e a capacidade de adaptação e aprendizagem das redes neurais (HAYKIN, 1994). Estes vêm sendo difundidos em diversos trabalhos em classificação (GONÇALVES et al., 2006), regressão (ISHIBUCHI; NII, 2001) e previsão de séries temporais (NAYAK et al., 2004). Alguns trabalhos que utilizam *Neuro-Fuzzy* podem ser observados em (KELLER; YAGER; TAHANI, 1992), (PEDRYCZ; ROCHA, 1993) e (JANG; SUN; MIZUTANI, 1997).

Dentre os sistemas *Neuro-Fuzzy*, um de grande relevância que pode ser usado com sucesso para problemas de previsão e aproximação de funções é o *Adaptative Neural Fuzzy Inference System* (*ANFIS*) (JANG, 1993), que utiliza a arquitetura tipo *Takagi-Sugeno*. Este modelo Neuro-*Fuzzy* tem uma gama de aplicações, dentre as quais se podem citar: previsão de nível de água em reservatórios (CHANG; CHANG, 2006), diagnóstico de doenças (POLAT; GÜNEŞ, 2007), previsão de séries temporais (ZOUNEMAT-KERMANI; TESHNEHLAB, 2008).

A arquitetura geral do modelo ANFIS está ilustrada na Figura 3.

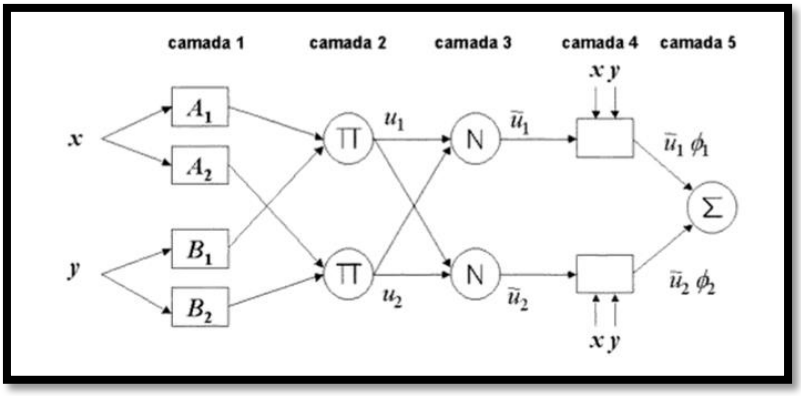

Figura 3-Modelo geral ANFIS

(REZENDE, 2003)

Seguindo cada uma das camadas, temos:

- Na camada 1 as suas saídas representam os graus de pertinência das entradas, tendo como base a premissa de cada entrada;
- Na camada 2 é calculado o grau de pertinência ao qual é submetido o consequente de cada uma das regras, onde cada neurônio desta camada executa uma operação *t-norm*;
- Na camada 3 as suas saídas representam a normalização dos valores de ativação de regras;
- Em 4 suas saídas são a contribuição de cada regra na saída total. Além disso, são calculadas as saídas através do grau de ativação consequente e o produto da saída normalizada da camada anterior;
- Por ultimo, temos a saída do sistema que pode ser calculada por um método de defuzzificação.

Os sistemas híbridos *Fuzzy-Evolucionário* tiveram grande importância para a realização deste trabalho, por isso, a seção 1.2 apresenta seus conceitos básicos.

1.2 Sistemas fuzzy evolucionários

A utilização de SFBR em conjunto com um AE teve início na década de 90, dando origem aos sistemas *Fuzzy* Evolucionários. O objetivo dessa concatenação é

aproveitar os pontos fortes das técnicas e tentar minimizar suas deficiências, visando à interpretabilidade que um SFBR pode proporcionar com a capacidade de otimização dos AE. Nesta linha de pesquisa podem-se citar alguns trabalhos, tais como: (HERRERA, 2008), (KARR, 1991),(THRIFT, 1991), (AMARAL et al., 2002), (FERREIRA; VELLASCO; TANSCHEIT, 2015).

A estrutura geral de um SFE pode ser observada na Figura 4. As Bases de regras (BR) são criadas a partir do Sistema de inferência e a Base de Parâmetros (BP) é composta pelos demais elementos como formato, número e disposição das funções de pertinência, universo de discurso, método de defuzzificação. A união da BR com BP é a denominada Base de Conhecimento (BC).

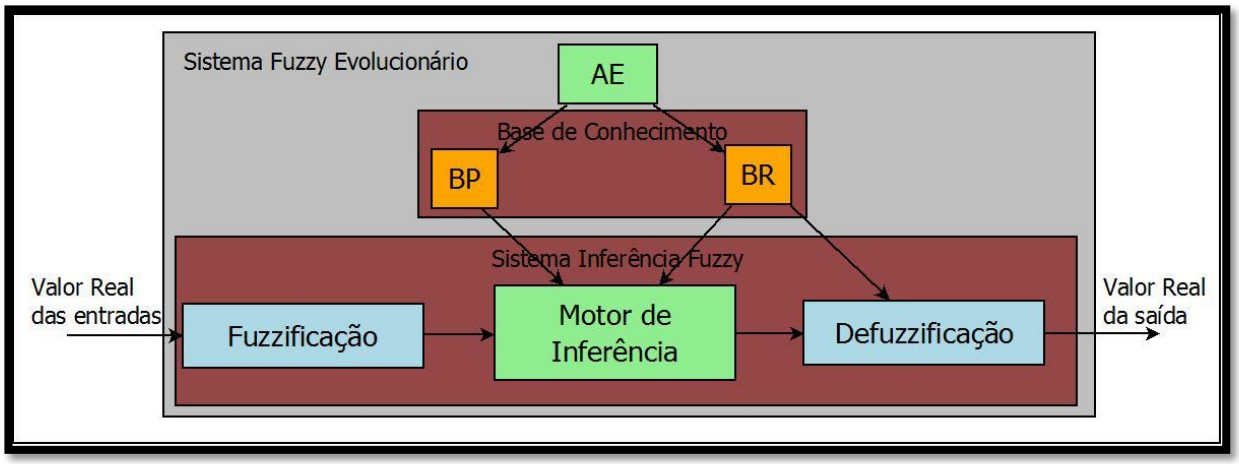

Figura 4-Modelo Sistema Fuzzy Evolucionário

Adaptado de (MERINO; TANSCHEIT; VELLASCO, 2015)

De modo geral, os SFEs atuam alterando a BP ou fazendo o aprendizado das BR, ou seja, atuam diretamente na BC.

Em relação à codificação das BRs em um SFE, existem quatro diferentes tipos. A primeira é denominada *Pittsburgh*, em que o indivíduo é representado por um conjunto de regras (FREITAS, 2003) e o melhor individuo é aquele que satisfaz os critérios definidos pelo usuário. No método *Michigan,* os indivíduos representam as regras e a população é resultado da concatenação de vários indivíduos. O método de avalição é complexo e indireto. Os sistemas baseados no método *Michigan* podem ser compostos por um sistema de inferência fuzzy (SIF), um mecanismo de geração de regras e um módulo de crédito que confere a

credibilidade de cada regra (HERRERA; MAGDALENA, 1997). A terceira proposta é o *Interative Rule Learning,* em que cada cromossoma é composto por indivíduo, o melhor individuo é geralmente escolhido como solução final (CORDÓN et al., 1999). A última proposta é o *Genetic Cooperative-Competitive Learning,* que opera de forma hibrida onde regras competem e cooperam entra si (HO et al., 2004).

2 Árvores de Padrões Fuzzy

Este capítulo tem por objetivo introduzir os conceitos básicos das Árvores de Padrões Fuzzy, mais precisamente abordando seus principais componentes, sua estrutura hierárquica e o método de aprendizado.

As APFs criadas por *Huang Gedeon* (HUANG; GEDEON; NIKRAVESH, 2008) e aperfeiçoadas por *Eike Hullermeier* (SENGE; HULLERMEIER, 2011), representam o conhecimento linguístico por meio de árvores, ao invés do uso tradicional de regras. Em sua estrutura hierárquica, as folhas são termos *fuzzy* associados aos atributos, e os nós são operadores utilizados em sistemas *fuzzy*. As saídas das árvores podem assumir qualquer valor y que é um grau de uma função de pertinência *fuzzy* G, cujo domínio é representado por Y. Existem dois modelos de árvores APFs criadas para o caso de regressão (SENGE; HÜLLERMEIER, 2010a). No primeiro a saída é representada por apenas uma árvore cujo valor é uma função, como por exemplo: equação linear (1) e equação sigmoide (2). No segundo modelo o valor de saída é representado por um processo de defuzzificação entre múltiplas árvores, cuja teoria será discutida na seção 2.1.3.

$$G: y \rightarrow \frac{y - a}{b - a} \tag{1}$$

$$G: y \rightarrow \frac{1}{1 + e^{-\propto y}} \tag{2}$$

A estrutura hierarquizada tem seu fluxo de informação de baixo para cima, ou seja, inicialmente os termos *fuzzy* associados aos atributos são inseridos nos pontos mais baixos, que são as folhas. Nos operadores as entradas são graus de

pertinências que podem ser provenientes das folhas ou do resultado de outros operadores. Em outras palavras, a junção de operadores e folhas forma uma APF, onde as entradas dos operadores podem ser termos fuzzy ou a saída de outros operadores. Por último, a saída é a junção dos nós, com os operadores no topo da estrutura.

Para um melhor entendimento, as próximas seções descrevem a estrutura e funcionamento das árvores, mais precisamente, os componentes das APFs e sua estrutura hierarquizada.

2.1 Componentes das APFs

2.1.1 Folhas

As entradas das árvores ou folhas representam o nível mais baixo por onde são inseridos os dados dos atributos, que precisam ser particionados em conjuntos *fuzzy*. Em outras palavras, tem-se uma base de dados representada pela equação (3):

$$T = \left\{x^{(i)}, y^{(i)}\right\}_{i=1}^{n} \subset X \times Y \tag{3}$$

Onde cada instância $\left\{x^{(i)}\right\}$ é um vetor

$$x \in X = X_1 \times X_2 \times \ldots \times X_n$$

X_i é o domínio do i-ésimo atributo A_i. Cada domínio é particionado em um conjunto de funções de pertinência $F_{i,j}$(4), que têm seus valores no intervalo de 0 a 1.

$$F_{i,j}: X_i \rightarrow [0,1] \qquad (\mathrm{j} = 1,2,\ldots,\mathrm{n}) \tag{4}$$

Em relação ao número de partições (granularidade), cada problema que se deseja resolver com as APFs pode apresentar uma granularidade mais adequada. No que se refere à forma das funções de pertinência, podem ser usadas todas aquelas que são normalmente utilizadas em sistemas fuzzy. Por exemplo, forma triangular com granularidade 3 representada na Figura 5.

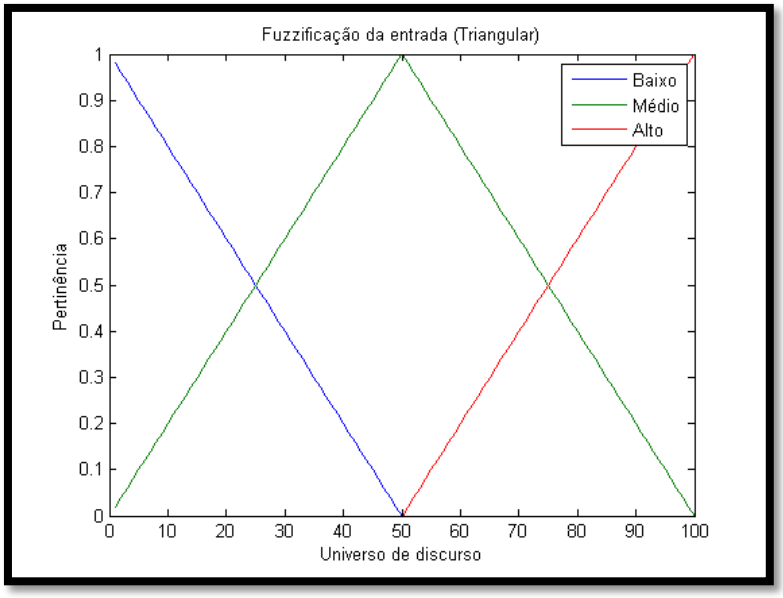

Figura 5-Exemplo fuzzificação (triangular)

As equações que representam as funções triangulares da Figura 5 podem ser facilmente implementadas, onde temos: (5) para o “baixo” ($F_{i,1}$); (6) para o “médio” ($F_{i,2}$) e (7) para o “alto” ($F_{i,3}$). O valor de *min* representa o menor valor do universo de discurso, o *max* é o maior valor e *c* representa o centro do universo de discurso.

$$F_{i,1}(x) = \begin{cases} 1 & x < \min \\ 0 & x > c \\ 1 - \dfrac{x - \min}{c - \min} & \text{outro caso} \end{cases} \tag{5}$$

$$F_{i,2}(x) = \begin{cases} 1 & x \leq \min \\ \dfrac{x - \min}{\text{centro} - \min} & \min < x < \text{centro} \\ 1 - \dfrac{x - \text{centro}}{\max - \text{centro}} & \text{centro} < x < \max \\ 0 & x \geq \min \end{cases} \tag{6}$$

$$F_{i,3}(x) = \begin{cases} 1 & x > \max \\ 0 & x < c \\ \dfrac{x - c}{\max - c} & \text{outro caso} \end{cases} \tag{7}$$

2.1.2 Operadores

Os operadores fuzzy que são utilizados nas APFS propostas por (SENGE; HÜLLERMEIER, 2010a) e também nesta dissertação são:

- *T-norm* e *T-conorm* (KLEMENT; MESIAR; PAP, 2013),
- *Weighted Average* (WA) e *Ordered Weighted Average* (OWA) (SCHWEIZER; SKLAR, 2011), (WITTEN; FRANK, 2005).

Todos os operadores utilizados nas APFs de regressão podem ser observados nas Tabelas 1, 2 e 3, onde, *a* e *b* são os valores de entrada dos nós. O valor de λ é um peso criado aleatoriamente entre os valores de 0 e 1.

Tabela 1-Operadores Fuzzy (*T-norm*)

Nº	T-norm	
1	Mínimo	$\min(a,b)$
2	Algébrico	ab
3	*Lukasiewicz*	$\max(a-1+b,0)$
4	Einstein	$\frac{ab}{2-(a+b-ab)}$

Tabela 2-Operadores Fuzzy (*T-conorm*)

Nº	T-conorm	
5	Máximo	$\max(a,b)$
6	Algébrico	$a+b-ab$
7	*Lukasiewicz*	$\min(a+b,1)$
8	Einstein	$\frac{a+b}{1+ab}$

Tabela 3-Operadores Fuzzy (Média)

Nº		Average
9	OWA	$\lambda * \mathrm{Max}\{a, b\} + (1 - \lambda) * \mathrm{Min}\{a, b\}$
10	WA	$\lambda * a + (1 - \lambda) * b$

2.1.3 Saídas das árvores

De acordo com (SENGE; HÜLLERMEIER, 2010a), o número de árvores é igual ao número de saídas, e cada saída é representada por um valor de pertinência. Quando existe apenas uma árvore, a relação entre entradas e saída pode ser mapeada de acordo com (8), onde G é um subconjunto fuzzy do domínio Y da variável de saída y.

$$f: (x_1, x_1, x_1, \dots, x_1) \rightarrow G(y) \tag{8}$$

O valor original do atributo de saída para este caso é encontrado através da inversa da função escolhida (9).

$$y = G^{-1}(z) \tag{9}$$

No caso de diversas partições *fuzzy* $G_1, G_2, G_3, \dots, G_n$, o mapeamento entre entradas e saídas pode ser representado através de (10).

$$f: (x_1, x_1, x_1, \dots, x_1) \rightarrow G_i(y) \tag{10}$$

Já o valor real é obtido por um dos diferentes métodos de defuzzificação disponíveis: centroide, altura, média dos máximos. (ROSS, 2009)

2.2 Estrutura hierárquica de uma APF

Este seção tem por objetivo descrever o de funcionamento de uma APF para o caso de regressão. A árvore na Figura 6 foi criada a partir de uma base de dados que tem por finalidade atribuir uma nota de 0/10 para certo tipo de vinho.

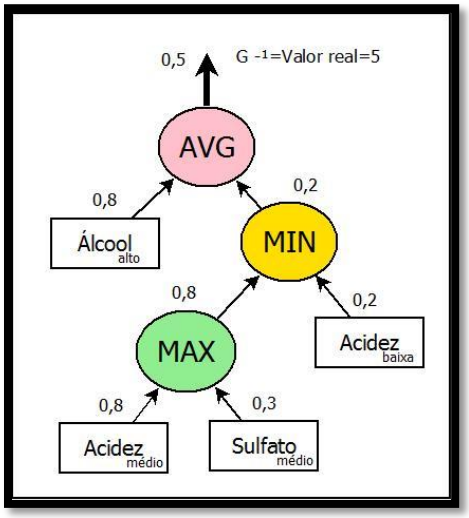

Figura 6-Exemplo APF qualidade de vinho

(adaptado de (SENGE; HÜLLERMEIER, 2010b))

Antes da inserção dos valores nas APFs é preciso transformar os valores dos atributos em graus de pertinência. Os atributos de entrada estão na Tabela 4, e a transformação do atributo álcool em termos *fuzzy* estão na Tabela 5. Após o processo de fuzzificação os graus de pertinência associados aos atributos são inseridos nas folhas. No ramo mais baixo o valor de $acidez_{médio}$ que é 0,8 e 0,3 do termo $sulfato_{médio}$ são utilizados como entradas do operador máximo, o valor encontrado é 0,8. O valor encontrado pelo operador máximo é entrada do operador mínimo em conjunto com o termo $acidez_{baixo}$ de valor 0,2, obtendo assim o valor de 0,2. Em seguida, a saída da árvore é encontrada através do operador média, cujas entradas são o termo o $álcool_{alto}$ com valor de 0,8 e a saída do operador mínimo que é 0,2. Assim, encontra-se o grau de pertinência de 0,5 que é a saída da árvore. A inversa da função (1) é qualidade do vinho, neste caso 5.

Tabela 4-BD vinhos (álcool antes da fuzzificação)

Álcool	Acidez	Sulfato	Qualidade
9,4	7,4	0,56	5
10	7,8	0,46	3
10,5	11,2	0,80	6
9,3	7,4	0,91	3

Fonte: (SENGE; HÜLLERMEIER, 2010b)

Tabela 5-BD vinhos (álcool após fuzzificação)

Álcool			Acidez	Sulfato	Qualidade
Baixo	Médio	Alto			
0,89	0,11	0,00	7,4	0,56	5
0,03	0,97	0,00	7,8	0,46	3
1	0	0	11,2	0,80	6
0	0	1	7,4	0,91	3

Fonte: (SENGE; HÜLLERMEIER, 2010b)

Normalmente, a síntese de APFs é realizada a partir dos dados de entrada e saída em uma estratégia de aprendizado supervisionado. A próxima seção descreve o método clássico de aprendizado para APFs de regressão.

2.3 Método de aprendizado

O método de aprendizado original proposto por (SENGE; HÜLLERMEIER, 2010b)para problemas de regressão é um variante do *Top-Down induction* (SENGE; HULLERMEIER, 2011), utilizado para problemas de classificação. O método realiza os passos descritos no pseudocódigo da Figura 7.

Para se encontrar a melhor árvore pelo método *Top-Down* utiliza-se o "*beam search*". De acordo com (SENGE; HULLERMEIER, 2011), o algoritmo *beam search* é um procedimento de busca tipo *best first search*. Apenas um número

predeterminado de melhores soluções é mantido como candidata. Em cada nível da árvore geram-se possíveis sucessores dos nós do nível atual, organizando-os em uma ordem crescente de custo computacional. Porém, para reduzir os requisitos de memória, este método só armazena um número pré-determinado, (largura do feixe de busca) de melhores sucessores de cada nível. Somente estes melhores sucessores serão expandidos futuramente.

1: {Inicializar}
2: $\boldsymbol{P} = \{A_{i,j}\}, i = 1, \dots, n; j = 1, \dots, m$
3: $\boldsymbol{C^0} = argmin\ B_{p \in P}[L(P)]$
4: $\varepsilon = 0.0025$
5: $t = 0$
6: {Indução}
7: {Iterações no Loop}
8: parar← falso
9: ***Enquanto*** não parar ***faça***
10: $l = l + 1$
11: $C^t = C^{t-1}$
12: {Loop em cada candidato}
13: ***Para todo*** $C_i^{t-1} \in C^{k-1}$ ***faça***
14: {Loop em cada folha do candidato escolhido}
15 ***Para todo*** $l_{escolhido} \in folhas\,(C_i^{t-1})$ ***faça***
16: {Loop em cada operador disponivel ψ}
17: ***Para todo*** $\psi \in folhas\ \Psi$ ***faça***
18: {Loop em quase todas árvores de padrões primitivas}
19: ***Para todo*** $P \in \setminus P\, l_{escolhido}$ ***faça***
20: $C^t = C^t \cup Substituirfolha\ (C_i^{t-1}, l_{escolhido}, \psi, P)$
21 ***Fim Para***
22: ***Fim Para***
23: ***Fim Para***
24: ***Fim Para***
25: $C^t = argmin\ B_{C_i^t \in C^t}[L(C_i^t)]$
26: **Se** $min_{C_i^t \in C^t}\,(L(C_i^t)) < (1+\varepsilon)\ min_{C_i^{t-1} \in C^{t-1}}\left(L(C_i^{t-1})\right)$ **então**
27: $parar = verdadeito$
28: ***Fim Se***
29: ***Fim Enquanto***
30: ***Retornar*** $argmin_{C_i^t \in C^t}\,(L(C_i^t))$

Figura 7-Método de aprendizado Top Down induction

Adaptado de (SENGE; HÜLLERMEIER, 2010b)

O método de criação *Top-Down* apresenta algumas deficiências:

- O algoritmo *beam search* pode ficar preso em sub ótimos globais, porque ele armazena apenas um número determinado de candidatos;
- Se a quantidade de atributos for muito grande, podem ocorrer problemas nas soluções através da maldição da dimensionalidade;
- Este critério pode apresentar *overfitting*, pois, como o critério de parada se baseia na melhoria do modelo a cada iteração, não se impõe um limite de crescimento da árvore.

Como objetivo de explorar um método alternativo ao descrito em(SENGE; HÜLLERMEIER, 2010b) para atenuar as deficiências encontradas no algoritmo original, este trabalho trata a síntese de APFs como um problema de otimização onde a técnica utilizada é a Programação Genética Cartesiana.

3 Programação Genética Cartesiana

Este capítulo tem como objetivo fornecer ao leitor o conhecimento básico para compreensão da Programação Genética Cartesiana. São abordados os conceitos de Algoritmos Evolucionários (AE), Programação Genética (PG) e Programação Genética Cartesiana (PGC).

3.1 Algoritmos evolucionários

Os AE são modelos computacionais que simulam a teoria da evolução das espécies através de seleção, mutação e reprodução. De acordo com *Linden* (LINDEN, 2006a), os AE funcionam mantendo uma população de estruturas, denominadas indivíduos ou cromossomos que se comportam de forma semelhante à evolução das espécies. A estas estruturas são aplicados os chamados operadores genéticos com recombinação e mutação. Cada indivíduo recebe uma avaliação que é uma quantificação da sua qualidade como solução do problema em questão. Com base nesta avaliação são aplicados os operadores genéticos de forma a simular a sobrevivência do mais apto.

Os AE são recomendados quando os algoritmos convencionais apresentam dificuldades de obter uma solução satisfatória (por exemplo, no caso de maximização de funções multimodais), sendo indicados, sobretudo, em situações onde o espaço de busca a ser pesquisado é extremamente grande. Uma característica interessante é que os AE são dependentes de fatores estatísticos (probabilísticos), tanto na fase da evolução quanto na fase de inicialização da

população. Portanto, trata-se de algoritmos que não encontram o melhor resultado em todas as suas execuções (LINDEN, 2006a), ou seja, os AE são heurísticas[2].

Dentre os AE, pode-se citar os algoritmos genéticos, a programação genética, as estratégias evolutivas e a programação evolutiva. Para maiores detalhes, os algoritmos descritos acima podem ser vistos em trabalhos, tais como: (GOLDBERG; HOLLAND, 1988), (MICHALEWICZ, 2013), (SCHWEFEL, 1993), (FOGEL, 2006), (KOZA, 1992a)e(BÄCK; SCHWEFEL, 1993).

3.2 Programação Genética

Um AE de grande importância para este trabalho é a Programação Genética (KOZA, 1990). A ideia é ensinar computadores a se programarem; a partir de especificações de comportamento, o computador deve ser capaz de induzir um programa que satisfaça o desejado (KOZA, 1992a).

A estrutura mais comum utilizada na PG é a de árvores sintáticas. Estas são compostas de funções e terminais, que determinam suas características e definem seu comportamento no ambiente (CASTRO; ZUBEN, 2003). As folhas são os terminais e podem ser variáveis ou constantes. Já os nós internos são compostos por funções. A Figura 8 representa um exemplo de equação ($y = \ln x_3 + ((x_1 * x_2) - 6)$). Na estrutura da árvore encontram-se os nós x_1,x_2e x_3 e, as folhas com as funções ln, subtração e produto. Além disso, existe a constante 6.

[2]Heurísticas são algoritmos polinomiais que não tem garantia nenhuma sobre a qualidade da solução encontrada, mas que usualmente tendem a encontrar a solução ótima ou ficar bem próximos dela.(LINDEN, 2006a)

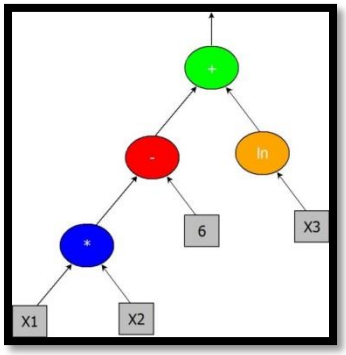

Figura 8-PG exemplo

De acordo com *Koza* (KOZA, 1992a), os programas criados apresentam uma variabilidade dinâmica que faz com que seja difícil especificar o tamanho e a forma da solução. Outra característica interessante é o fato de a PG não necessitar de pré-processamento das entradas e pós-processamento das saídas. A evolução do programa ocorre dentro do domínio do problema e as saídas são expressas no mesmo conjunto, não sendo necessários processos de tradução ou mapeamento. A respeito da criação das árvores, existem diversos métodos, no entanto, devido à sua simplicidade serão descritos os métodos *Full*, G*row* e *Ramped Half-Half*.

O método *Full* escolhe aleatoriamente apenas um conjunto de não terminais até uma profundidade “D-1”. Isso faz com que todas as árvores sejam uniformes e tenham uma profundidade D (GRINGS, 2006). A Figura 9 demonstra o processo de criação de uma árvore utilizando o *Full*. Observa-se que as folhas estão na mesma profundidade, o que não implica que todas as árvores tenham o mesmo número de nós ou o mesmo formato. (POLI et al., 2008)

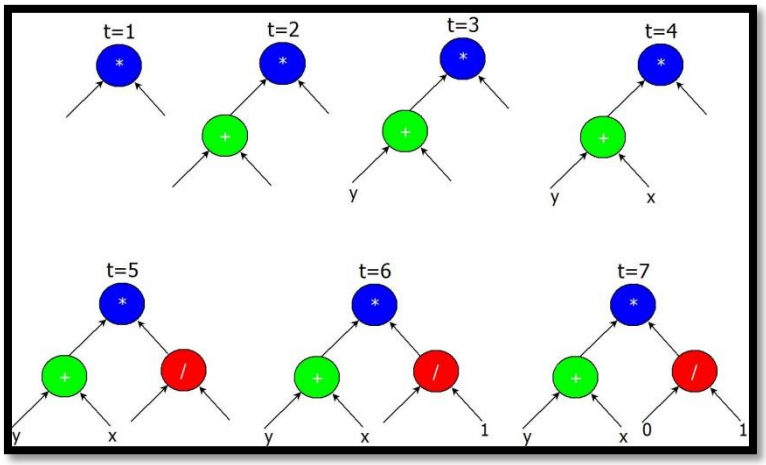

Figura 9-PG método *Full*

No método *Grow,* a criação de árvores pode apresentar profundidade variável. Os nós são escolhidos de forma aleatória entre funções e terminais, respeitando-se uma profundidade máxima (MAIA, 2005). A Figura 10 apresenta a criação de uma árvore pelo método *Grow* (POLI et al., 2008).

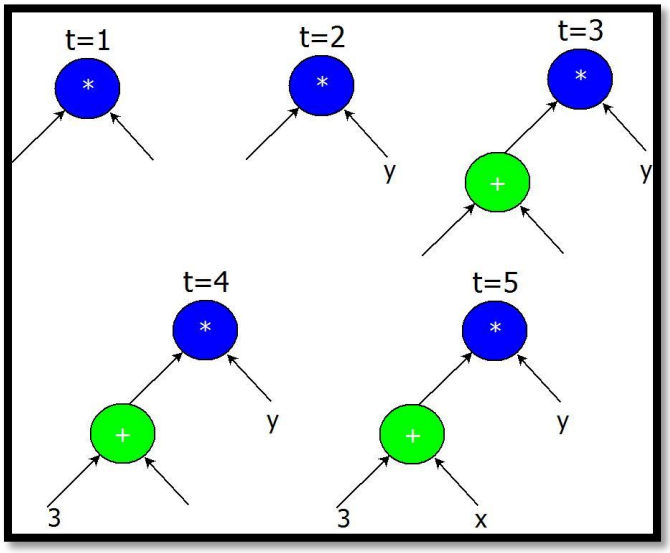

Figura 10-PG Método *Grow*

O *Ramped Half-Hal f*é na verdade uma combinação dos métodos *Grow* e *Full*, ou seja, cada método é responsável pela criação da metade da população. Este método não precisa que a população inicial seja totalmente aleatória e a profundidade das árvores pode ser variável (KOZA, 1992b).

Na PG, como em qualquer AE, a seleção dos indivíduos ocorre de maneira probabilística com base na aptidão. Em outras palavras, o indivíduo que apresentar melhor aptidão tem a probabilidade maior de ser selecionado. O método de seleção mais utilizado na PG é o *torneio,* devido a sua eficiência e simplicidade. Esta seleção preserva melhor a diversidade da população, porque a escolha é realizada apenas comparando o valor da função de custo entre os indivíduos que participam do torneio. Neste método, existe um parâmetro denominado tamanho do torneio (K), que define quantos indivíduos são selecionados aleatoriamente dentro da população para competir. Uma vez definidos os competidores, aquele que possuí a melhor avaliação é selecionado para a aplicação do operador genético (LINDEN, 2006b).

O operador de cruzamento mais utilizado é o *subtree crossover*. Em dois indivíduos são selecionados um nó em cada. No primeiro descarta-se uma das sub-

árvores do nó selecionado, logo em seguida, ocorre a substituição do nó que foi selecionado na segunda árvore (MARQUES, 2013). Para um melhor entendimento, o processo citado acima pode ser observado na Figura 11.

Na mutação, o operador mais utilizado é o *point mutation.* Proposto por *McKay, Willis e Barton* (MCKAY; WILLIS; BARTON, 1995), o método substitui um nó qualquer da árvore por outro, aleatoriamente criado. Os descendentes possuem o mesmo tamanho dos pais. Para o caso de nós função, acrescenta a restrição de trocá-lo por outro com a mesma quantidade de argumentos, garantindo assim a integridade da árvore.

Um ponto negativo da PG é que ela não se mostrou satisfatória para grandes problemas, devido ao denominado *Bloat* (SOULE, 1998), que ocorre devido à tendência dos programas gerados com a PG ficarem muito maiores que o funcionalmente necessário. Este crescimento, além de criar um código que não influencia significativamente o desempenho, dificulta a busca por um código que vá realmente melhorar os programas. Pesquisas sobre o *Bloat* na PG indicam que este crescimento ocorre em qualquer técnica evolucionária que possua representações de tamanho variável. (SOULE; HECKENDORN, 2002)

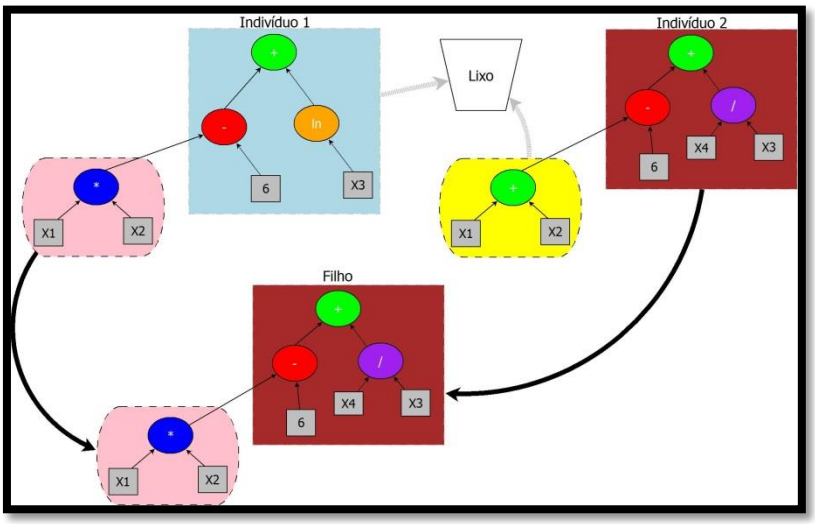

Figura 11-Crossover Subtree

Algumas extensões da PG são *Parallel Distributed Genetic Programing (PDGP)* (POLI, 1996), *Linear Genetic Programming* (LGP) (BRAMEIER; BANZHAF, 2007) e *Cartesian Genetic Programming* (Programação Genética Cartesiana-PGC).

Esta última teve grande papel no desenvolvimento deste trabalho e, por isso, a próxima seção aborda seus conceitos básicos.

3.3 Conceitos sobre a Programação genética Cartesiana

A PGC (MILLER, 2011) é uma variante da PG aplicada neste trabalho para a construção das APFs em problemas de regressão. Estas representam programas escritos na forma de grafos, que são codificados em uma sequência linear de inteiros, representados por uma grade de nós computacionais de n-dimensões, como observado na Figura 12.

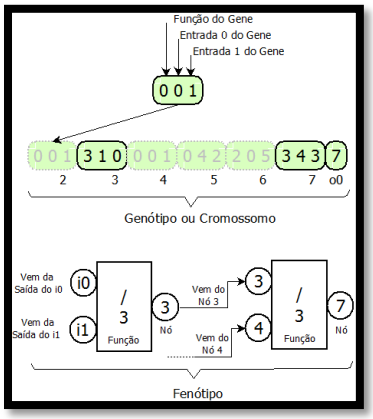

Figura 12-Representação de Genótipos e Fenótipos

Adaptado de (MILLER, 2011)

Os genótipos observados na Figura 12 representam a sequência linear de números inteiros, onde cada número inteiro é chamado de gene, que pode ser rotulado de função, conexão ou saída. Cada nó é representado pela junção destes três tipos de genes, dos quais o gene de função utiliza os genes de conexão para gerar uma saída através de um conjunto de funções previamente definido. Alguns exemplos de funções básicas podem ser encontrados na Tabela 6. A entrada de cada nó pode ser uma entrada do sistema ou a saída de um nó da coluna anterior, porém nunca da mesma coluna ou da coluna à frente.

Tabela 6-Exemplo Genes de Função

Referência do Gene	Função
0	+
1	-
2	*
3	/

Existem alguns parâmetros que devem ser definidos pelo usuário, tais como: *level-back*, aridade, número de colunas e número de linhas. O primeiro diz ao gene de conexão o número máximo de colunas anteriores às quais ele pode se conectar. Já o segundo é responsável pela quantidade de entrada dos nós, com mostra a Figura 13 com aridade 2 e 3. Os últimos dois são responsáveis pelo formato da grade. Modificando estes últimos parâmetros, podem se formar variados tipos de representações de nós, como mostra os casos a e b na Figura 14.

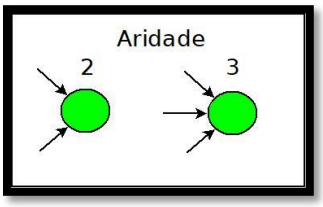

Figura 13-PGC aridade

O processo evolutivo ocorre nos genótipos, sendo necessária uma decodificação destes em fenótipos, cuja formatação está no domínio da solução do problema. Quando o genótipo é decodificado alguns nós podem ser desativados. O genótipo tem tamanho fixo e o fenótipo pode ter tamanho variado.

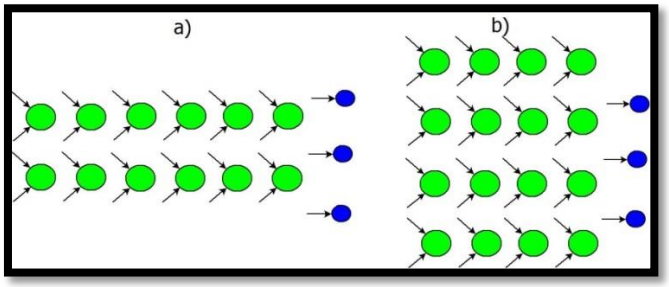

Figura 14-Exemplo de configurações de PGC

Adaptado de (MILLER, 2011)

Na PGC o operador de mutação tem importante papel, sendo bastante comum apenas a sua utilização. Normalmente, um gene é escolhido aleatoriamente e o seu valor é modificado para um valor válido. Tanto genes de função como de conexão e saída podem ser escolhidos para a mutação. A quantidade de genes que podem ser alterados a cada aplicação do operador de mutação está ligada à denominada taxa de mutação. Observando a Figura 15, que utilizou o conjunto de funções da Tabela 6, nota-se como o genótipo e o fenótipo podem ser alterados após um processo de mutação. Na primeira parte, a codificação do genótipo formou um fenótipo com saída em 6. Após um processo de mutação, o fenótipo da Figura 15 (b) é totalmente diferente daquela da Figura 15 (a), inclusive sua a saída foi alterada.

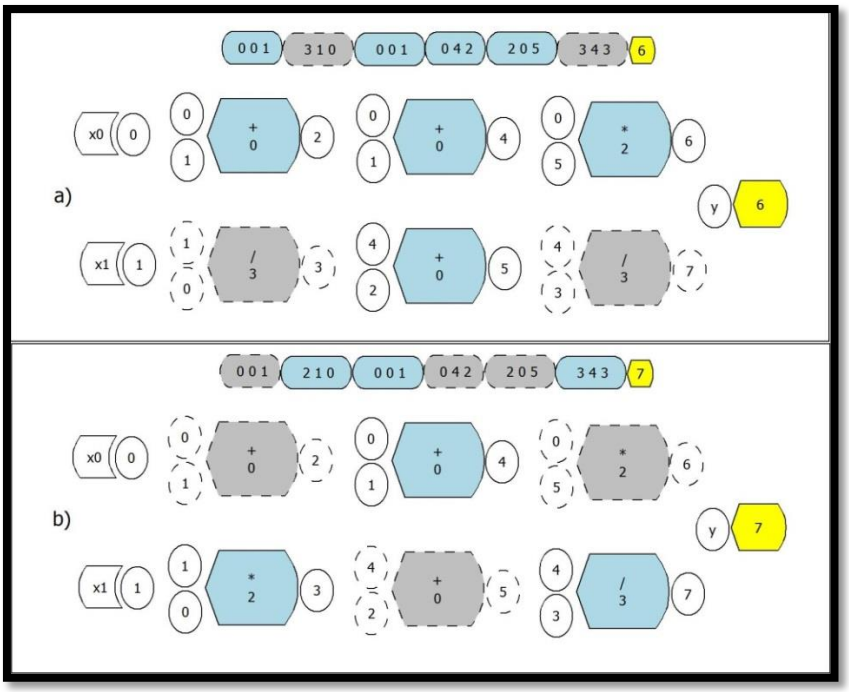

Figura 15-Processo de mutação

O *crossover* é pouco usado na PGC. O operador de um ponto foi usado, mas verificou-se ser prejudicial para os subgrafos dentro do genótipo e para o desempenho (LIU et al., 2011). A estratégia evolutiva mais utilizada é a (1 + λ). Normalmente, λ é igual 4. Primeiramente, são criados cinco indivíduos aleatoriamente e o mais apto, ou seja, aquele com o melhor valor da função de aptidão, é selecionado para a geração seguinte, e os quatro restantes são excluídos.

O operador de mutação é aplicado quatro vezes no indivíduo com menor aptidão da geração anterior, formando uma nova população. Novamente, o operador de mutação é aplicado quatro vezes no mais apto da geração anterior. O processo continua até se atingir o critério de parada. Entretanto, caso dois ou mais indivíduos tenham a mesma aptidão, o último indivíduo criado vai para a próxima geração.

A Figura 16 mostra um exemplo da estratégia evolutiva (1 + λ).

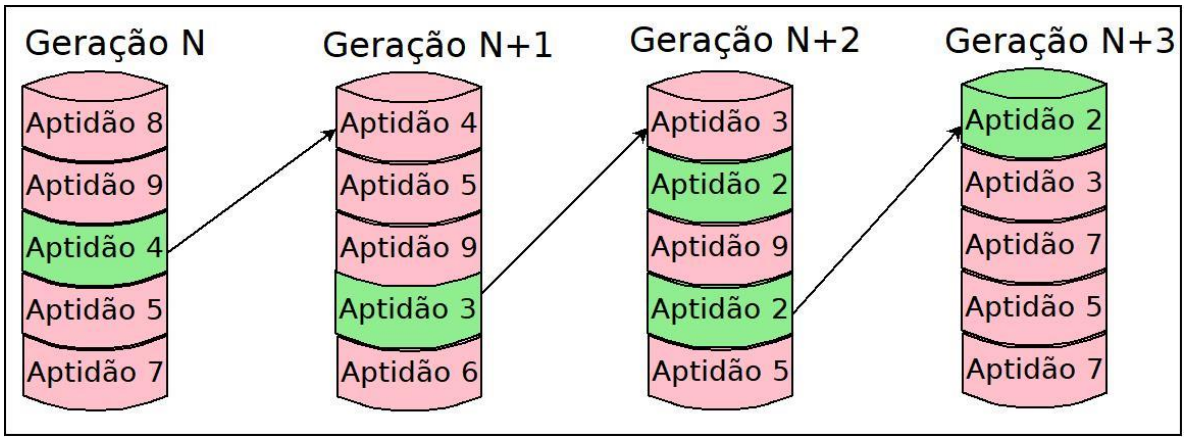

Figura 16-Exemplo de estratégia evolutiva

4 Modelo proposto (APFR-PGC)

Este capítulo apresenta os modelos propostos neste trabalho, descrevendo toda a estrutura hierárquica que compõe as árvores de padrões fuzzy, desde a preparação dos bancos de dados até a apresentação do resultado final. A explicação dos modelos aqui propostos está dividida em seções, que descrevem: a visão geral dos modelos, as partições fuzzy e o treinamento.

4.1 Visão geral

Como mencionado na introdução, este trabalho tem como finalidade explorar um método alternativo aos tradicionais SFBR. Para tal, foi criado um sistema híbrido que utiliza as estruturas denominadas Árvores de Padrões Fuzzy, sendo estas sintetizadas através da PGC. Dentre os motivos que levaram à escolha das APFs, podem-se citar:

- Serem atraentes do ponto de vista de interpretabilidade, uma vez que se consegue observar uma expressão que leva à tomada de decisão;
- Seleção automática dos atributos de entrada. Os melhores atributos são escolhidos automaticamente pelo modelo sem necessidade de o usuário fazer explicitamente esta seleção;
- A forma gráfica. Pode-se observar qual termo tem um peso maior na estrutura hierárquica da árvore.

Em relação à estratégia de aprendizado, o método original denominado *beam search* pode levar à escolha de uma solução sub-ótima, pois, a exploração do espaço de busca é feita de forma “gulosa”. Já a PGC pode explorar grandes espaços de busca de forma eficiente, o que se reflete em chances maiores de se encontrar uma solução ótima. Outro aspecto positivo é que a PGC tem uma

representação baseada em grafos, o que permite que as APFs sejam representadas de forma eficiente.

De forma geral os modelos aqui propostos obedecem à estrutura descrita no diagrama de blocos da Figura 17. As principais diferenças entre os modelos são descritas no decorrer deste capítulo.

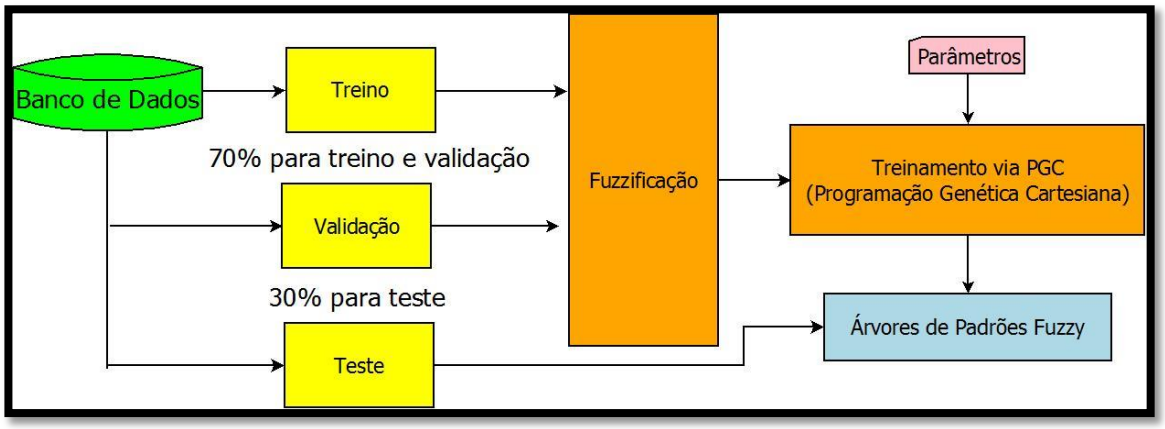

Figura 17-Diagrama de blocos dos modelos APFs para regressão

Inicialmente, o banco de dados é dividido em treinamento, validação e teste. Após esse particionamento, os dados são fuzzificados e usados no treinamento dos genótipos da PGC. Com o intuito de se evitar *overfitting* utilizou-se o conjunto de validação com parada antecipada. Após esta fase, utiliza-se o conjunto de teste para obter uma medida de generalização das APFs sintetizadas.

4.2 Partições fuzzy

Como dito anteriormente, o banco de dados foi dividido em treinamento, validação e teste. Em seguida, os dados são fuzzificados para serem usados nas APFs. Nos modelos propostos neste trabalho, os atributos de entrada são particionados em conjuntos fuzzy. Já a saída é particionada em número igual ao de árvores que se deseja criar.

4.2.1 Partição das entradas

Esta etapa tem relação direta sobre como os dados são inseridos nas folhas. Em relação aos dois modelos, existem três fatores que devem ser levados em consideração: a forma das funções de pertinência, o rótulo linguístico e o suporte de cada função.

Na forma das funções de pertinência, pode-se usar qualquer forma geralmente utilizada para sistemas fuzzy (triangular, gaussiana, trapezoidal, etc), entretanto, neste trabalho, optou-se por utilizar as formas triangular e gaussiana. A primeira foi escolhida devido à sua simplicidade, por apresentar informação satisfatória acerca do termo linguístico e fazer uma distribuição linear entre os termos limites dos conjuntos (PEDRYCZ, 1994). A gaussiana foi usada pelo fato de ser bastante utilizada em problemas de aproximação de funções (CALLEGARI-JACQUES, 2009). Para os rótulos linguísticos, foram utilizados conjuntos de três (‘baixo’, ‘médio’ e ‘alto’), cinco (‘muito baixo’, ‘baixo’, ‘médio’, ‘alto’ e ‘muito alto’) e sete (‘extremamente baixo’, ‘muito baixo’, ‘baixo’, ‘médio’, ‘alto’, ‘muito alto’ e ‘extremamente alto’). Idealmente, para se ter uma interpretabilidade maior, árvores mais compactas e um ajuste melhor do suporte que represente de forma precisa os rótulos linguísticos é necessário um especialista. Como tal profissional qualificado nem sempre se encontra disponível, optou-se por empregar funções ditas uniformes (ou forte) particionadas e o particionamento *Tukey*. Esta última, que vem sendo utilizada em alguns trabalhos (MERINO; TANSCHEIT; VELLASCO, 2015), (FERREIRA; VELLASCO; TANSCHEIT, 2015) e (LEGARDA; VELLASCO; TANSCHEIT, 2016), faz uso da informação estatística dos quartis: 0° quartil (valor mínimo), 1°quartil, 2° quartil (mediana), 3° quartil e 4° quartil (valor máximo). A Figura 18 apresenta o tipo de função de pertinência com 5 rótulos linguísticos do tipo triangular uniforme.

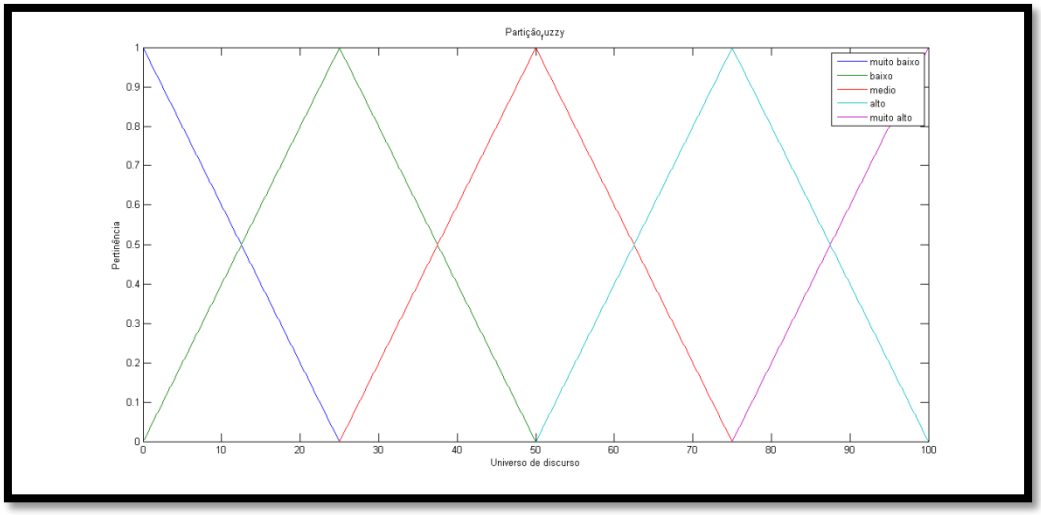

Figura 18-Função de pertinência uniforme

A Figura 19 apresenta um exemplo de *Tukey* triangular com 5 rótulos linguísticos.

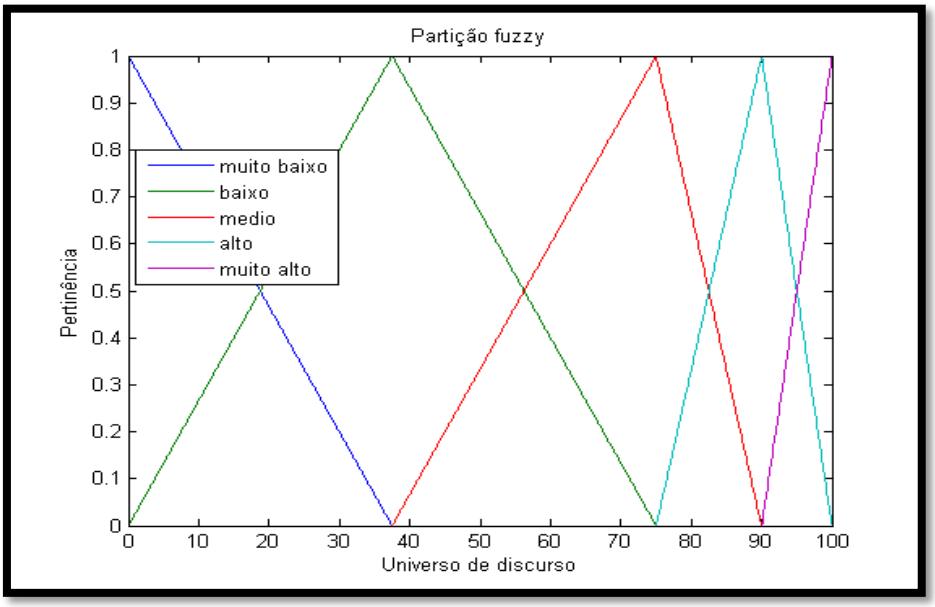

Figura 19-Função de pertinência *Tukey*

Além dos tipos de configurações usualmente utilizadas, novas configurações foram utilizadas na busca de resultados que também se mostrassem satisfatórios, então após vários experimentos preliminares optou-se por utilizar os conjuntos de entrada mostrados na Figura 20. A esta configuração deu-se o nome 3A.

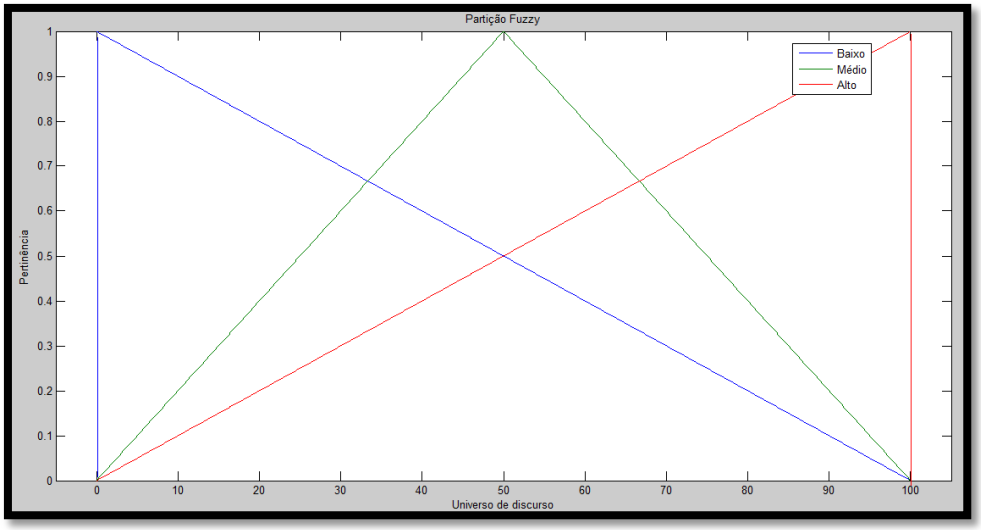

Figura 20-Função de pertinência 3A

4.2.2 Partição da saída

O número de partições fuzzy de saída está ligado diretamente ao número de árvores que se deseja criar. Ressalte-se que, igualmente à subseção anterior, se deve levar em conta a forma das funções de pertinência, o rótulo linguístico e o suporte de cada função. Neste trabalho, cada modelo apresenta peculiaridade em sua saída.

O primeiro é representado por uma única função de pertinência; para este modelo foi utilizada a equação (1), onde, os parâmetros “a” e “b” são respectivamente os limites mínimo e máximo do conjunto de suporte. Para encontrar o valor real de resposta neste modelo deve-se calcular G^{-1}, ou seja, a equação inversa.

O segundo modelo utiliza diversas árvores, onde cada uma fornece o grau de pertinência para seus respectivos conjuntos. De posse destes graus e dos conjuntos, é possível obter a saída resultante através de um processo de defuzzificação (centroide, altura,...), como proposto originalmente pelo método de *Senge* e *Hüllermeir* (SENGE; HÜLLERMEIER, 2010a). Neste trabalho foram utilizadas funções triangulares e o método da altura pela simplicidade computacional.

4.3 Treinamento através da PGC

A fase de treinamento é responsável por encontrar APFs que consigam expressar um valor de resposta. Antes do treino propriamente dito, é de extrema importância definir alguns parâmetros, tais como: quantidade de gerações, tamanho da população, a função *aptidão*, taxa de mutação, o tamanho da população, número de linhas, colunas e o *levelback*. Em relação à estratégia evolutiva, adotou-se a $1 + \lambda$, descrita na seção 3.3. Outro parâmetro bem importante, que relaciona diretamente APF e PGC, é o conjunto de funções da PGC. Em outras palavras, o conjunto de funções é responsável pelos operadores fuzzy definidos nas Tabelas 1, 2 e 3.

Outro estudo realizado que atuou modificando a natureza dos conjuntos *fuzzy* foi a inserção dos *modificadores* nas APFs. Para tal, foram adicionados ao conjunto de funções PGC o intensificador muito ($\mu(x)^2$), o diluidor pouco ($\mu(x)^{1/2}$) e o complemento ($\overline{\mu(x)}$). Outro aspecto interessante foi a possibilidade de modificar a aridade das árvores. O exemplo da Figura 21 apresenta uma árvore aleatória escolhida após uma determinada geração em *a*. Após o processo de mutação mostrado na parte *b*, o nó representado pela função *WA* (em vermelho) é substituído pelo intensificador. Então em *b* um nó foi escolhido para ser excluído. Em *c,* tem-se a árvore final.

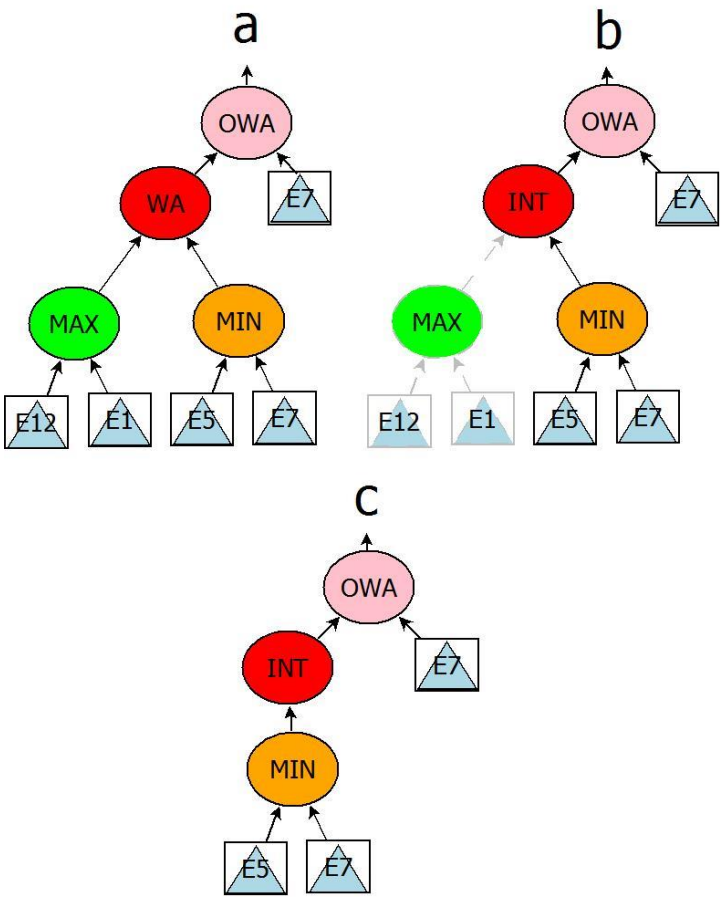

Figura 21-Árvore:a)sem diluidor b) com diluidor

4.3.1 Estrutura PGC e APFs

Como explicado no capítulo 4, a PGC é representada por uma sequência linear de inteiros denominada de genótipo (MILLER, 2011), onde cada inteiro é um gene e cada nó é a junção dos genes de ligação e função, como mostrado na Figura 12.

Com o intuito de melhor representar APFs, cada nó é formado por uma sequência de três números inteiros. Existem dois nós de ligação e um de função (operadores fuzzy). Porém, nem sempre se usam todos os nós de um genótipo; na verdade, os nós utilizados nas APFs criadas pela PGC são a decodificação dos genótipos (fenótipos). Foram desenvolvidos dois tipos diferentes de representação desses genótipos, que variam dependendo do modelo. As Figuras 22 e 23 mostram a relação entre genótipos e fenótipos na criação das APFs para cada um dos

modelos aqui desenvolvidos, onde os genes de função utilizados são os da Tabela 1, 2 e 3 e o número de entradas possíveis (termos linguísticos) é 11.

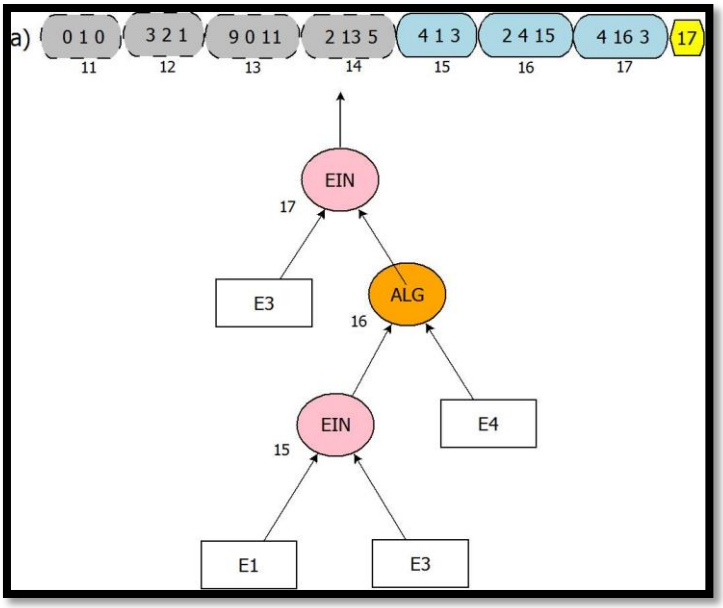

Figura 22-CGP APF modelo 1

A Figura 22 representa o modelo 1, onde o genótipo tem 7 nós e uma saída. A codificação para encontrar a resposta utiliza 3 nós (15, 16 e 17). Estes nós, quando conectados, formam uma APF, cuja saída está em 17. As folhas são os termos fuzzy dos atributos e os nós, operadores *fuzzy,* que utilizam como entrada folhas ou a saída de outros nós. O fluxo da informação caminha de baixo para cima, até chegar ao topo onde a saída é um grau de pertinência da função (1). O valor real é encontrado através da inversa da função de saída escolhida.

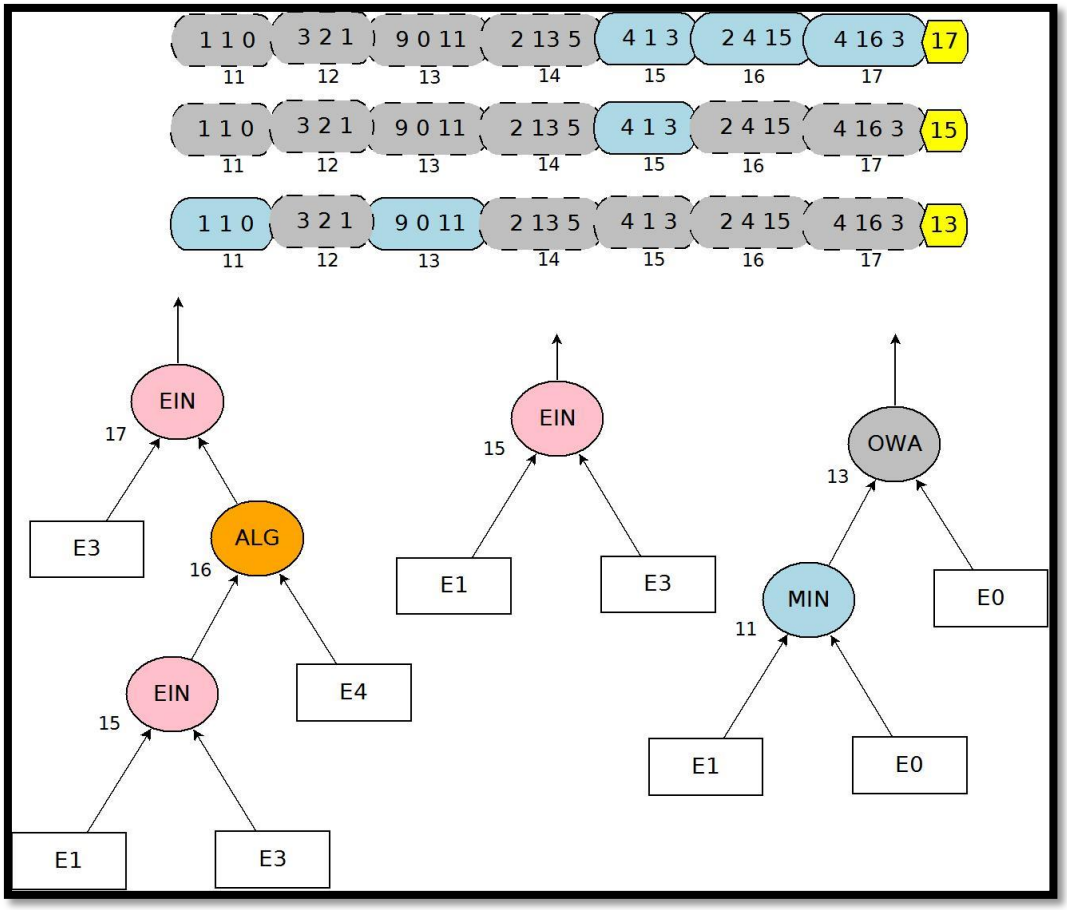

Figura 23-CGP APF modelo 2

No modelo 2 (Figura 23), diferentemente do primeiro modelo, existem múltiplos genótipos que são codificados criando tantas saídas quanto o número de árvores. Na Figura 23 existem 3 genótipos que, quando codificados, formam APFs, onde a saída em 13 utiliza os nós 11 e 13; a APF com saída em 15 utiliza unicamente o nó 15; e a última árvore é formada pelos nós 15, 16 e 17, sendo a saída em 17. O tratamento das entradas, operadores e o fluxo de informação é igual ao do modelo 1. Cada saída também é representada por valores entre [0,1], que representam cada uma das funções triangulares fortemente particionadas. O valor real de saída é encontrado através do método de defuzzificação da altura.

4.3.2 Métodos de avaliação

Para encontrar o erro de cada árvore de regressão, diversas métricas podem ser utilizadas, tais como *Mean Square Error* (MSE), *Root Mean Square Error* (RMSE), *Mean Absolute Error* (MAE), *Mean Absolute Percentage Error* (MAPE), entre outras. Diferentemente do trabalho de (SANTOS; DO AMARAL, 2014), que divide a *aptidão* em uma parte para similaridade e outra limitando o número de genes, no caso da regressão utiliza-se apenas uma métrica.

Em relação às métricas de desempenho, optou-se pelo MSE (11) na fase de treinamento e pelo RMSE (12) na fase de teste, igualmente ao utilizado em (SENGE; HÜLLERMEIER, 2010b), onde y_t é o valor de resposta da APF e $f_{j,t}$ é o valor esperado, para ambos os modelos.

$$\mathrm{MSE}_j = \frac{1}{T}\sum_{t=1}^{T}(y_t - f_{j,t})^2 \tag{11}$$

$$\mathrm{RMSE}_j = \frac{1}{T}\sqrt{\sum_{t=1}^{T}(y_t - f_{j,t})^2} \tag{12}$$

4.3.3 Critério de parada

Para escolher o melhor ponto em que o AE deve parar a sua execução foram utilizados três diferentes critérios. No primeiro, a parada ocorre após um número predeterminado de gerações. No segundo, quando após um determinado número gerações houver um piora na métrica de erro para o conjunto de validação. O último é executado no caso de a aptidão não se alterar após um número predeterminado de gerações no treinamento e validação.

5 Estudo de Caso

Este capítulo apresenta diversos experimentos realizados para avaliar o desempenho dos modelos propostos para a tarefa de regressão. Os experimentos foram divididos em duas partes: obtenção dos parâmetros default e comparação com outros modelos. Na primeira parte experimental foi feito um procedimento para obtenção dos parâmetros *default* dos modelos aqui propostos. Já na segunda, ocorre a comparação com modelos clássicos da literatura e também com o modelos originais das APFs.

As bases de dados utilizadas em ambas as partes dos experimentos foram retiradas do repositório de *KEEL* (“KEEL: A software tool to assess evolutionary algorithms for Data Mining problems (regression, classification, clustering, pattern mining)”, 2016, p. KEEL), da *University of California Irvine* (UCI) (“UCI Machine Learning Repository: Data Sets”, 2015.]) e da (“Weka 3 - Data Mining with Open Source Machine Learning Software in Java”, [s.d.]). Com o intuito de mesclar BDs utilizadas em (SENGE; HÜLLERMEIER, 2010b) com outros BDs, a Tabela 7 contém os dados utilizados para a para obtenção dos parâmetros *default*. Já em relação à comparação com outros modelos foram usados os BD da Tabela 8.

Tabela 7- Bases de dados para melhor configuração default

	Banco de dados	Abreviação	Pontos	Atributos
1	Auto MPG8	MPG8	398	8
2	Clevelend	CLE	303	13
3	Housing	HOU	506	14
4	Hungarian	HUN	294	14
5	Pharynx	PHA	195	12
6	Eficiência energética 1	EFC1	768	9
7	Eficiência energética 2	EFC2	768	9
8	Winered	WIN	1600	12
9	Slump	SLU	103	11
10	Concrete Compressive strengh	CON	1030	9

Tabela 8-Bases de dados para comparação entre modelos

	Banco de dados	Abreviação	Pontos	Atributos
1	Auto MPG6	MPG6	398	6
2	Cholesterol	CHO	303	14
3	Daily electricityenergy	DEE	365	7
4	Fruitfly	FRU	125	5
5	Power Linear	PWL	200	11
6	Quake	QUA	2178	4
7	Sensory	SEM	576	12
8	Stock	STO	950	10
9	Strike	STR	626	7
10	Wine White	WiW	4898	12

Para verificar se existe ou não diferença estatisticamente significativa nos resultados experimentais, sempre que possível foi utilizado o teste de *Friedman* (DERRAC et al., 2011).

Em todos os experimentos foi utilizada a técnica de validação cruzada com 5 pastas. Em relação aos parâmetros da PGC foram utilizadas 1000 colunas, com o intuito de que o espaço de busca da PGC pudesse compreender um grande número de possibilidades entre operadores e termos fuzzy. O número linhas utilizado foi igual a1. O *Levelback* de 999 buscou dar ao algoritmo a possibilidade de explorar a ligação com todos os nós dos genótipos criados. A estratégia evolutiva $1 + 4$, onde λ = 4 definiu o tamanho da população sendo igual a 5. Em experimentos preliminares se mostrou satisfatório utilizar o número de gerações de 5000 e a taxa de mutação igual a 0,1. A função de aptidão utilizada foi o MSE, a mesma utilizada para criar as APFs em (SENGE; HÜLLERMEIER, 2010b).

5.1 Ajuste dos parâmetros

Esta seção tem por objetivo avaliar qual configuração é mais vantajosa para cada um dos modelos. O delineamento experimental sequencial aqui proposto seguiu a ordem da Tabela 9. Os resultados das tabelas de experimentos são a média de 30 experimentos e seu desvio padrão. Já a métrica de desempenho utilizada foi o RMSE.

Tabela 9- Ajuste de parâmetros experimentos

Nº	Experimento
1	Número de termos linguísticos de entrada
2	Tipo de Particionamento das entradas
3	Formato das funções de entrada
4	Redução de operadores
5	Número de termos linguísticos de Saída
6	Modificadores

5.1.1 Número de termos linguísticos de entrada

Na configuração das entradas observou-se que número de termos linguísticos (granularidade), formato e tipos das funções de pertinência são mais vantajosos para as entradas de cada um dos modelos.

No primeiro experimento, que examinou a granularidade dos modelos, foram utilizados conjuntos com no máximo 7 termos linguísticos (com objetivo de diminuir a complexidade computacional e aumentar a interpretabilidade). Mais precisamente foram usados modelos com granularidade de 3, 5 e 7. Outro experimento consistiu na comparação com o conjunto 3A, descrito na seção 4.2.1.

Os resultados de ambos os modelos para granularidades 3, 5 e 7 constam nas Tabelas 10 e 11, respectivamente.

Tabela 10-Número de partições de entrada (modelo 1)

	3	5	7
BD	Erro±Desv	Erro±Desv	Erro±Desv
EFC1	**3,99±0,37**	4,30±0,34	4,06±0,48
EFC2	4,01±0,36	4,06±0,34	**3,99±0,40**
CON	**11,36±1,16**	12,23±1,06	13,14±0,88
HOUS	**4,76±0,56**	5,38±0,47	6,01±0,80
MPG8	**3,76±0,96**	5,99±2,09	7,25±2,67
CLE	**0,85±0,06**	0,86±0,09	0,85±0,09
HUN	**0,48±0,09**	0,50±0,12	0,54±0,14
PHA	355,88±69,51	**334,46±46,77**	370,35±35,33
SLU	**3,16±1,14**	3,68±1,51	4,07±1,60
WIN	**0,77±0,02**	0,78±0,04	0,79±0,02

Tabela 11-Número de partições de entrada (modelo 2)

	3	5	7
BD	Erro±Desv	Erro±Desv	Erro±Desv
EFC1	**3,93±0,43**	4,72±0,67	4,29±0,91
EFC2	**3,97±0,60**	4,31±0,39	4,34±0,31
CON	**12,34±1,66**	13,05±1,46	13,47±1,05
HOUS	**5,24±0,61**	5,75±0,72	6,59±0,76
MPG8	3,73±0,38	5,68±0,97	**2,24±0,40**
CLE	0,85±0,05	0,85±0,08	**0,85±0,85**
HUN	0,46±0,04	**0,41±0,04**	0,46±0,03
PHA	**290,11±30,05**	299,50±36,12	306,09±11,21
SLU	**4,58±1,41**	5,85±1,30	5,88±1,08
WIN	0,78±0,02	**0,77±0,04**	0,77±0,02

A partir dos resultados acima, a partição com granularidade 3 foi a escolhida, poiso teste de *Friedman* comprovou que no modelo 1 houve diferença estatisticamente significativa entre as partições. No modelo 2 também optou-se pela granularidade 3, que teve um desempenho superior em 60% dos bancos aqui testados.

Outro experimento realizado foi a comparação das partições 3 com o 3A. Os resultados podem ser observados nas Tabelas 12 e 13.

Tabela 12-Granularidade 3 e 3A das entradas (modelo 1)

		3	3A
Nº	BD	Erro±Desv	Erro±Desv
1	EFC1	3,99±0,37	**3,48±0,42**
2	EFC2	4,01±0,36	**3,52±0,30**
3	CON	11,36±1,16	**10,45±0,87**
4	HOUS	4,76±0,56	**4,62±0,32**
5	MPG8	3,76±0,96	**3,45±0,90**
6	CLE	0,85±0,06	**0,80±0,04**
7	HUN	**0,46±0,09**	0,47±0,09
8	PHA	355,88±69,51	**342,54±80,82**
9	SLU	**3,16±1,14**	4,62±1,48
10	WIN	**0,77±0,02**	0,77±0,03

Tabela 13-Granularidade 3 e 3A das entradas (modelo 2)

		3	3A
Nº	BD	Erro±Desv	Erro±Desv
1	EFC1	**3,93±0,43**	4,10±0,54
2	EFC2	**3,97±0,60**	4,04±0,26
3	CON	12,34±1,66	**11,99±0,55**
4	HOUS	5,24±0,61	**5,13±0,89**
5	MPG8	**3,73±0,38**	3,88±0,79
6	CLE	0,85±0,05	**0,81±0,04**
7	HUN	0,46±0,04	**0,43±0,04**
8	PHA	**290,11±30,05**	298,47±31,24
9	SLU	4,58±1,41	**4,49±0,93**
10	WIN	**0,78±0,02**	0,79±0,04

Analisando ambas as tabelas, o conjunto escolhido foi o 3A, pois foi vitorioso em 70% dos bancos no modelo 1. No teste estatístico não se observou diferença estatisticamente significativa.

5.1.2 Tipo de particionamento das entradas

Seguindo o processo de obtenção dos parâmetros *default* dos modelos APFR-PGC, foram analisados os conjuntos do tipo 3A e *Tukey*. Os resultados desse experimento podem ser observados nas Tabelas 14 e 15.

Tabela 14-Tipo de partição das entradas (modelo 1)

		3A	*Tukey*
Nº	BD	Erro±Desv	Erro±Desv
1	EFC1	**3,48±0,42**	4,44±1,44
2	EFC2	**3,52±0,30**	4,84±0,38
3	CON	**10,45±0,87**	10,59±1,08
4	HOUS	**4,62±0,32**	4,70±0,51
5	MPG8	**3,45±0,90**	4,52±1,87
6	CLE	**0,80±0,04**	0,85±0,15
7	HUN	0,47±0,09	**0,45±0,08**
8	PHA	342,54±80,82	**316,76±67,35**
9	SLU	4,62±1,48	**4,58±0,98**
10	WIN	0,77±0,03	**0,77±0,02**

Tabela 15--Tipo de partição das entradas (modelo 2)

		3A	*Tukey*
Nº	BD	Erro±Desv	Erro±Desv
1	EFC1	**4,10±0,54**	5,21±1,57
2	EFC2	**4,04±0,26**	5,23±0,51
3	CON	**11,99±0,55**	11,87±1,07
4	HOUS	**5,13±0,89**	5,47±0,47
5	MPG8	3,88±0,79	4,05±2,12
6	CLE	**0,81±0,04**	0,94±0,31
7	HUN	**0,43±0,04**	0,45±0,08
8	PHA	298,47±31,24	**274,58±22,42**
9	SLU	**4,49±0,93**	4,91±0,89
10	WIN	0,79±0,04	**0,79±0,02**

Nos resultados das tabelas acima o teste de *Friedman* mostrou que não houve diferença estatisticamente significativa entre as amostras. Optou-se pelo pela partição 3A, pois esta foi vencedora em um maior número de banco de dados.

5.1.3 Formato das funções de entrada

Neste experimento foram analisadas funções de entrada para os conjuntos ditos fortemente particionados. Os testes se limitaram as funções do tipo triangular e gaussiano.

Tabela 16-Formato das funções de entradas (modelo 1)

		Triangular	*Gaussiana*
Nº	BD	Erro±Desv	Erro±Desv
1	EFC1	**3,48±0,42**	4,02±0,42
2	EFC2	**3,52±0,30**	3,83±0,31
3	CON	**10,45±0,87**	12,65±0,92
4	HOUS	**4,62±0,32**	4,68±0,39
5	MPG8	**3,45±0,90**	4,36±1,54
6	CLE	**0,80±0,04**	0,86±0,07
7	HUN	0,47±0,09	**0,45±0,15**
8	PHA	**342,54±80,82**	354,92±66,07
9	SLU	**4,62±1,48**	4,74±1,28
10	WIN	**0,77±0,03**	0,77±0,04

Tabela 17-Formato das funções de entradas (modelo 2)

		Triangular	*Gaussiana*
Nº	BD	Erro±Desv	Erro±Desv
1	EFC1	4,10±0,54	**3,99±0,38**
2	EFC2	**4,04±0,26**	4,08±0,46
3	CON	**11,99±0,55**	12,80±1,36
4	HOUS	5,13±0,89	**5,07±0,36**
5	MPG8	3,88±0,79	**3,71±0,35**
6	CLE	**0,81±0,04**	0,81±0,03
7	HUN	**0,43±0,04**	0,48±0,13
8	PHA	**298,47±31,24**	309,95±59,16
9	SLU	**4,49±0,93**	5,54±1,18
10	WIN	0,79±0,04	**0,79±0,01**

Analisando o resultado desse experimento, observou-se que no modelo 1 a função triangular foi melhor em 90% dos casos, enquanto no modelo 2, em 60%. O teste estatístico mostrou que houve diferença estatisticamente significativa.

5.1.4 Número de árvores de padrões fuzzy

Este experimento teve o objetivo de analisar o impacto do aumento no número de APFs. Devido ao aumento do esforço computacional, a granularidade foi limitada para conjuntos de 3, 5 e 7 árvores. Os resultados desse experimento podem ser observados na Tabela 18.

Tabela 18-Número de APFs (modelo 2)

		3	5	7
Nº	BD	Erro±Desv	Erro±Desv	Erro±Desv
1	EFC1	**4,10±0,54**	4,62±0,51	5,06±0,44
2	EFC2	**4,04±0,26**	4,04±0,32	5,32±0,42
3	CON	**11,99±0,55**	12,67±0,79	13,00±0,58
4	HOUS	**5,13±0,89**	5,64±0,55	6,71±0,46
5	MPG8	3,88±0,79	5,21±2,74	4,47±0,57
6	CLE	**0,81±0,04**	0,88±0,13	0,94±0,07
7	HUN	0,43±0,04	**0,41±0,02**	0,42±0,02
8	PHA	298,47±31,24	**294,73±30,13**	305,55±28,89
9	SLU	**4,49±0,93**	5,88±0,97	5,99±0,64
10	WIN	0,79±0,04	0,76±0,01	**0,76±0,01**

O teste de *Friedman* mostrou que houve diferença estatisticamente significativa. Optou-se pela partição de granularidade 3, que proporcionou um melhor resultado, como observado na Tabela 19, que apresenta o *rank* médio.

Tabela 19-Média dos *ranks do* número de árvores

Partições	3	5	7
Média dos *ranks*	**1,55**	1,80	2,65

5.1.5 Redução do número de operadores

Seguindo a sequência experimental, com o intuito de obter uma melhor interpretabilidade das APFs, foi analisado o desempenho em termos de acurácia para um conjunto reduzido de operadores *fuzzy*. Foi utilizado todo o conjunto de

operadores contidos nas Tabelas 1, 2 e 3 (vide Capítulo 3) contra o conjunto reduzido de operadores da Tabela 20.

Tabela 20-Conjunto reduzido de operadores

Nº	**Operadores**	
1	Mínimo	$\min(a, b)$
2	Máximo	$\max(a, b)$
3	OWA	$\lambda * \mathrm{Max}\{a, b\} + (1 - \lambda) * \mathrm{Min}\{a, b\}$
4	WA	$\lambda * a + (1 - \lambda) * b$

Os resultados desse experimento podem ser observados na Tabela 21 e Tabela 22.

Tabela 21-Redução de Operadores (modelo 1)

		Conj. Normal	Conj. Reduzido
Nº	BD	Erro±Desv	Erro±Desv
1	EFC1	**3,48±0,42**	4,35±0,81
2	EFC2	**3,52±0,30**	4,55±1,29
3	CON	**10,45±0,87**	13,87±2,13
4	HOUS	**4,62±0,32**	5,24±1,01
5	MPG8	**3,45±0,90**	5,26±1,58
6	CLE	**0,80±0,04**	0,84±0,10
7	HUN	0,47±0,09	**0,45±0,07**
8	PHA	342,54±80,82	**337,79±62,51**
9	SLU	**4,62±1,48**	5,65±2,24
10	WIN	**0,77±0,03**	0,96±0,38

Tabela 22--Redução de Operadores (modelo 2)

		Normal	Reduzido
Nº	BD	Erro±Desv	Erro±Desv
1	EFC1	4,10±0,54	**3,96±0,46**
2	EFC2	4,04±0,26	**3,99±0,20**
3	CON	**11,99±0,55**	12,93±0,91
4	HOUS	**5,13±0,89**	6,32±1,19
5	MPG8	3,88±0,79	**3,74±0,32**
6	CLE	**0,81±0,04**	0,86±0,07
7	HUN	0,43±0,04	**0,41±0,03**
8	PHA	**298,47±31,24**	340,09±101,17
9	SLU	**4,49±0,93**	5,51±1,29
10	WIN	0,79±0,04	**0,77±0,03**

De forma geral, o conjunto reduzido de operadores teve resultados similares ao do conjunto de todos os operadores, o que é confirmado pelo teste de *Friedman* que revelou não haver diferença estatisticamente significativa nos resultados. Optou-se pelo conjunto com todos os operadores, pois este em media conseguiu resultados de acurácia menores.

5.1.6 Modificadores

Para a execução desse experimento foram adicionadas novas funções na tabela PGC, com o intuito de simular os modificadores propostos por *Zadeh*: o intensificador muito ($\mu(x)^2$), o diluidor pouco ($\mu(x)^{1/2}$) e o complemento ($\overline{\mu(x)}$).

O resultado dos experimentos pode ser observado nas Tabelas 23 e 24.

Tabela 23-Modificadores (modelo 1)

		$\overline{modificadores}$	$modificadores$
Nº	BD	Erro±Desv	Erro±Desv
1	EFC1	3,48±0,42	**3,43±0,53**
2	EFC2	**3,52±0,30**	3,63±0,37
3	CON	**10,45±0,87**	10,82±1,07
4	HOUS	**4,62±0,32**	4,67±0,37
5	MPG8	3,45±0,90	**3,32±0,54**
6	CLE	**0,80±0,04**	0,83±0,07
7	HUN	0,47±0,09	**0,43±0,05**
8	PHA	**342,54±80,82**	381,33±93,30
9	SLU	4,62±1,48	**4,45±1,08**
10	WIN	**0,77±0,03**	0,77±0,03

Tabela 24-Modificadores (modelo 2)

		$\overline{modificadores}$	$modificadores$
Nº	BD	Erro±Desv	Erro±Desv
1	EFC1	**4,10±0,54**	4,17±0,65
2	EFC2	4,04±0,26	**3,85±0,20**
3	CON	11,99±0,55	**11,47±0,95**
4	HOUS	**5,13±0,89**	5,24±0,73
5	MPG8	3,88±0,79	**3,25±0,23**
6	CLE	**0,81±0,04**	0,87±0,07
7	HUN	**0,43±0,04**	0,44±0,05
8	PHA	**298,47±31,24**	323,14±73,73
9	SLU	**4,49±0,93**	5,02±1,35
10	WIN	0,79±0,04	**0,77±0,01**

Os valores das tabelas dos experimentos mostram que em ambos os modelos uso de *modificadores* não melhorou significativamente o desempenho das APFs, o que pode ser comprovado pelo teste estatístico.

5.2 Comparação de modelos

Esta segunda parte experimental tem por objetivo fazer os experimentos observados na Tabela 25.

Tabela 25- Comparação de modelos experimentos

Nº	Experimentos (comparação)
1	Comparação de algoritmos
2	Comparação entre APFs
3	Comparação de profundidade
4	Comparação de avaliações

Os parâmetros da PGC e das APFs são os mesmos definidos na primeira parte experimental. Os bancos de dados utilizados nos experimentos constam na Tabela 8.

5.2.1 Comparação de algoritmos

Esta seção tem o objetivo de mostrar a comparação dos modelos aqui propostos com modelos na literatura. São usados os seguintes algoritmos: Máquinas de vetores de suporte (*Support Vector Machies Regression-SVMR),* Regressão linear *(Linear Regression-LR*), K vizinhos mais próximos *(*K-*NearestNeighbors-KNN),* Árvores de decisão (Regression trees-*REPTree)* e a Perceptron multicamadas (M*ultilayer Perceptron*-MLP).

As SVMR foram introduzidas por Vapnik (CORTES; VAPNIK, 1995), apresentando seus fundamentos na teoria de aprendizado estatístico, tendo

aplicação para problemas de regressão e classificação ((DE MORAES LIMA, 2004), (CHAVES; VELLASCO; TANSCHEIT, [s.d.])). A sua formulação engloba o princípio de minimização de risco estrutural (VAPNIK; VAPNIK, 1998) (SRM-*structural risk minimization*). Assim, o SVMR faz a minimização de um limite superior sobre o erro de generalização. Então, máquinas que usam o princípio do SVM tendem a ter uma boa generalização em dados não observados. No caso de classificação, as SVMs constroem hiperplanos que tem por objetivo posicionar as amostras o mais longe possível do hiperplano ótimo. Já no caso de regressão, a ideia é forçar as amostras a ficar o mais próximo possível do hiperplano ótimo.

O modelo de LR é um dos modelos mais utilizados devido a sua simplicidade; seu principal objetivo é obter uma relação linear entre as variáveis dependentes com independentes. No caso de haver uma variável dependente e uma independente, tem-se a regressão linear simples. Por outro lado, quando são incorporadas várias variáveis independentes, tem-se a regressão linear múltipla.

O KNN é um algoritmo elegante e simples utilizado em problemas de regressão, classificação e em reconhecimento de padrões (KUNCHEVA, 2004). O KNN propõe um aprendizado baseado em instâncias, ou aprendizado preguiçoso, significando que na fase de aprendizagem, ele simplesmente armazena um conjunto de instâncias rotuladas (conjunto de treinamento).

No modelo de árvores de decisão, a construção é feita de forma recursiva, de cima para baixo (JIAWEI; KAMBER, 2001). O conjunto de treinamento é dividido de acordo com um teste sobre uma das variáveis independentes, formando-se regiões distintas e não sobrepostas. Todo ponto do conjunto de treinamento que estiver em uma mesma região terá a mesma previsão, que será a média dos valores da variável dependente dos pontos que estiverem nesta região. O objetivo é encontrar um conjunto de regiões (hiperparalelogramos) que minimize a soma quadrática dos resíduos (HASTIE; TIBSHIRANI, 1990).

De acordo com (HAYKIN, 1994), as Redes Neurais Artificiais (RNA) são um sistema maciçamente paralelo e distribuído, composto por unidades de processamento simples que possuem uma capacidade natural de armazenar e utilizar conhecimento. Elas são consideradas modelo do tipo *caixa preta*, o que não

permite saber como a decisão é tomada. A RNA utilizada neste trabalho é a MLP. O treinamento da MLP originalmente é feito pelo algoritmo *Backpropagation*.

Todos os modelos utilizaram a ferramenta *WEKA ou Matlab*. Como ajuste dos parâmetros iniciais, foi utilizado o *default* oferecido pelo programa, exceto para o caso da KNN em que o número K foi limitado em 5.

Os resultados obtidos de RMSE e seus respectivos desvios padrões são mostrados na Tabela 26.

Tabela 26-Comparação de modelos

	LR	MLP	SVMR	KNN	REPTree	APF1	APF2
	Erro	Erro	Erro	Erro	Erro	Erro	Erro
MPG6	3,43±0,38	**3,20±0,55**	3,53±0,48	4,00±0,33	3,44±0,36	3,30±0,36	3,46±0,55
CHO	51,47±6,91	103,65±19,49	52,46±7,39	53,60±4,17	53,18±7,13	**49,78±6,03**	49,28±6,77
DEE	0,41±0,04	0,47±0,07	0,41±0,04	0,44±0,03	0,50±0,05	**0,41±0,04**	0,46±0,05
FRU	16,07±3,37	19,83±4,83	16,22±3,65	17,95±2,55	15,78±3,20	14,82±3,58	**14,68±4,42**
PWL	2,23±0,28	2,31±0,32	**2,22±0,30**	2,41±0,29	2,07±0,28	3,26±0,42	3,35±0,40
QUA	0,19±0,01	0,21±0,03	0,20±0,01	0,20±0,01	**0,19±0,01**	0,19±0,02	0,19±0,02
SEN	**0,77±0,03**	1,31±0,11	0,79±0,04	0,78±0,03	0,77±0,06	0,83±0,05	0,84±0,04
STO	2,34±0,11	1,23±0,15	2,40±0,12	**0,81±0,04**	1,24±0,14	2,63±0,36	3,47±0,49
STR	478,28±193,50	552,39±230,62	**470,20±205,07**	477,00±109,98	497,90±186,36	529,52±180,03	547,19±210,54
WIW	0,75±0,02	0,79±0,09	0,76±0,02	0,82±0,01	**0,75±0,02**	0,94±0,07	0,92±0,06

A respeito de ambos os modelos aqui propostos, o desempenho encontrado foi satisfatório, seus resultados mostraram-se competitivos em relação aos demais modelos de comparação. O teste de *Friedman* mostrou que não houve diferença estatisticamente significativa entre os modelos. A Tabela 27 expressa o *rank* médio, o que deixa clara a competitividade de desempenho em relação aos outros modelos.

Tabela 27-Rank médio modelos de comparação

LR	MLP	SR	KNN	REPTree	APF1	APF2
2,50	5,200	3,70	4,50	3,20	4,10	4,80

5.2.2 Comparação entre APFs

Seguindo a sequência experimental, a seção faz a comparação dos modelos aqui propostos contra os modelos PTTD reg e o PTTD system, criados por (SENGE; HÜLLERMEIER, 2010a), descritos no capítulo 0. Os resultados de acurácia entre a estratégia T*op Down* e a PGC estão ilustrados na Tabela 28.

Tabela 28-Comparação Top Down e PGC (acurácia)

	PTTD reg	PTTD system	APF1	APF2
BD	Erro±Desv.	Erro±Desv.	Erro±Desv	Erro±Desv.
MPG6	3,59±0,29	3,76±0,15	**3,30±0,36**	3,46±0,55
CHO	52,19±6,02	52,88±6,94	**49,78±6,03**	49,28±6,77
DEE	0,44±0,04	0,42±0,05	**0,41±0,04**	0,46±0,05
FRU	17,34±3,24	16,21±3,59	14,82±3,58	**14,68±4,42**
PWL	3,38±0,34	**2,83±0,34**	3,26±0,42	3,35±0,40
QUA	0,20±0,02	0,19±0,03	**0,19±0,02**	0,19±0,02
SEM	**0,82±0,05**	0,86±0,04	0,83±0,05	0,84±0,04
STO	2,85±0,55	2,71±0,43	**2,63±0,36**	3,47±0,49
STR	548,62±176,54	**493,58±195,61**	529,52±180,03	547,19±210,54
WiW	**0,82±0,01**	0,83±0,02	0,94±0,07	0,92±0,06

Os resultados mostram que, em média, o modelo APF1 leva uma ligeira vantagem em relação aos modelos originais, o que pode ser observado pelo *rank* da

Tabela 29. O teste de *Friedman* mostrou que não houve diferença estatisticamente significativa entre os resultados.

Tabela 29-Rank médio entre APFs

APF1	APF2	PTTD reg	PTTD system
1,80	2,700	3,00	2,30

5.2.3 Comparação de profundidade

Esta seção tratada interpretabilidade. Foi analisada a menor profundidade alcançada pelos modelos APF1 e PTTD reg. Os resultados da comparação desta seção estão descritos na Tabela 30.

Tabela 30-comparação de profundidade (modelo 1)

BD	PTTD reg	APF1
MPG6	3	3
CHO	**2**	4
DEE	3	**2**
FRU	3	3
PWL	1	1
QUA	2	2
SEN	1	1
STO	3	3
STR	2	2
Win	3	3

Os resultados mostram que, de forma geral, os modelos das APF aqui criados conseguiram criar árvores com profundidades iguais às da PTTDreg.

5.2.4 Comparação de avaliações

Nesta comparação observou-se a quantidade necessária de avaliações necessárias para se obter os resultados do experimento anterior. Foram comparados o PTTDreg e APF1. Para o APF1 foi feito o cálculo da quantidade de avaliação para o pior caso possível (sem a parada antecipada). Os resultados estão mostrados na Tabela 31.

Tabela 31-Quantidade de Avaliações

BD	PTTD reg	APF1
MPG6	29K	2,9k
CHO	106,48K	3,3
DEE	42,12k	3,5k
FRU	18,72k	3,1k
PWL	1k	2,8k
QUA	5,67k	2,6k
SEN	10,84k	3,5k
STO	50,93k	3,3k
STR	22,68k	3,3k
WiN	141,57k	3,9k

Nesta comparação nota-se que, de uma forma geral, os modelos PTTDreg necessitam de um número maior de avaliações em média para se alcançar uma solução satisfatória, o que pode ser observado na vitória em 90% dos BD no modelo APF1. Na utilização da estratégia *Top-down,* pode ocorrer uma explosão de possibilidades quando o número de atributos for muito grande, isto não ocorre na PGC, assim a programação genética cartesiana consegue chegar a resultados equivalentes ao da estratégia *Top-down* com um número menor de avaliações.

6 Conclusão

Atualmente, o volume de dados armazenado continua crescendo rapidamente. Infelizmente, devido à dificuldade do ser humano de interpretar tamanha quantidade de dados, muita informação e conhecimento podem estar sendo desperdiçados, ficando ocultos nas Bases de Dados. Neste contexto, surgiu a área multidisciplinar chamada “mineração de dados”. Um paradigma bem atraente, que consegue traduzir em termos matemáticos informações imprecisas é a teoria dos conjuntos *fuzzy.* Sua representação do conhecimento é feita de uma forma atraente, pois a forma linguística que ela fornece é facilmente compreensível e interpretável. Uma alternativa aos SFBR são as APFs, que, em relação a interpretabilidade, são mais vantajosas quando comparadas a métodos tipo *caixa preta*.

Neste trabalho com o intuito de explorar um método alternativo aos SFBR criou-se um sistema hibrido *fuzzy* evolucionário. Este utiliza a estrutura hierárquica Árvores de Padrões Fuzzy para resolver problemas de regressão. O algoritmo de aprendizado original utilizado foi substituído pela Programação Genética Cartesiana, que é um método de busca global capaz de explorar grandes espaços de forma eficiente e em que a representação dos programas na forma de grafos pode ser facilmente utilizada para representar APFs.

Criaram-se dois modelos que aproveitam a estrutura principal da CGP denominada genótipo. No modelo1, a estrutura é formada por um único genótipo. Já no modelo 2, foram utilizados múltiplos genótipos.

No estudo de caso buscou-se a melhor configuração para as APFs, analisar o impacto de *modificadores* e a redução dos operadores. Além disso, foi feita a comparação com modelos clássicos de resolução de problemas de regressão. Também houve a comparação com o modelo original analisando acurácia, interpretabilidade e número de avalições.

De forma geral, os modelos aqui propostos mostraram-se competitivos em relação aos modelos clássicos. Na comparação com os modelos de (SENGE; HÜLLERMEIER, 2010b), os resultados das APFs aqui propostas mostram ligeira vantagem em acurácia nos bancos de dados testados. Na questão da interpretabilidade, os modelos conseguiram representar o conhecimento de forma similar às APFS originais. Já em relação ao número de avaliações, os modelos aqui propostos necessitaram deum número menor de avalições. Isso mostra que a estratégia PGC aqui utilizada consegue varrer bancos de dados de uma forma global não sofrendo, assim, da “maldição da dimensionalidade”.

As propostas para trabalhos futuros estão listadas a seguir:

- Realizar uma extensão do modelo para resolver problemas de previsão de series temporais;
- Analisar o impacto da Multichromosome Cartesian Genetic Programming (MC-CGP) (MILLER, 2011) na criação de modelos de APFs com múltiplas árvores ou em banco de dados que apresentem mais de uma saída;
- Incorporar a idéia de *Bagging* (BREIMAN, 1996) e *Boosting* (FREUND; SCHAPIRE, 1995) para criar novos modelos;
- Aperfeiçoar a criação das APFs através de algoritmos multiobjetivos.

REFERÊNCIAS

AMARAL, J. F. M. et al. **Evolutionary fuzzy system design andimplementation**. Neural Information Processing, 2002. ICONIP'02. Proceeding sof the 9th International Conference on. **Anais**...IEEE, 2002. Disponível em: <http://ieeexplore.ieee.org/abstract/document/1198998/>. Acesso em: 29 jan. 2017

ASAI, H. T.-S. U.-K. Linear regression analysis with fuzzy model. **IEEE Trans. Systems Man Cybern**, v. 12, p. 903–907, 1982.

BÄCK, T.; SCHWEFEL, H.-P. An overview of evolutionary algorithms for parameter optimization. **Evolutionary computation**, v. 1, n. 1, p. 1–23, 1993.

BRAMEIER, M. F.; BANZHAF, W. **Linear genetic programming**. [s.l.] Springer Science & Business Media, 2007.

BREIMAN, L. Bagging predictors. **Machine learning**, v. 24, n. 2, p. 123–140, 1996.

CALLEGARI-JACQUES, S. M. **Bioestatística: princípios e aplicações**. [s.l.] Artmed Editora, 2009.

CASTRO, L. N. DE; ZUBEN, F. J. V. **Computação Evolutiva: Uma Abordagem Pragmática**. [s.l: s.n.].

CHANG, F.-J.; CHANG, Y.-T. Adaptive neuro-fuzzy inference system for prediction of water level in reservoir. **Advances in Water Resources**, v. 29, n. 1, p. 1–10, 2006.

CHAVES, A.; VELLASCO, M.; TANSCHEIT, R. EXTRAÇÃO DE REGRAS FUZZY DE MÁQUINAS VETOR SUPORTE PARA CLASSIFICAÇÃO EM MÚLTIPLAS CLASSES. [s.d.].

CORDÓN, O. et al. MOGUL: a methodology to obtain genetic fuzzy rule-based systems under the iterative rule learning approach. **International Journal of Intelligent Systems**, v. 14, n. 11, p. 1123–1153, 1999.

CORTES, C.; VAPNIK, V. Support-vector networks. **Machine learning**, v. 20, n. 3, p. 273–297, 1995.

DE MORAES LIMA, C. A. **Comitê de Máquinas: uma abordagem unificada empregando máquinas de vetores-suporte**. [s.l.] Universidade Estadual de Campinas, 2004.

DERRAC, J. et al. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. **Swarm and Evolutionary Computation**, v. 1, n. 1, p. 3–18, 2011.

FERREIRA, M.; VELLASCO, M. M. B. R.; TANSCHEIT, R. **GPFIS-Forecast: Um Sistema Fuzzy Genético Baseado em Programação Genética Multigênica para Problemas de Previsão Univariada**. [s.l.] PUC-Rio, 2015.

FOGEL, D. B. **Evolutionary computation: toward a new philosophy of machine intelligence**. [s.l.] John Wiley & Sons, 2006. v. 1

FREITAS, A. A. A survey of evolutionary algorithms for data mining and knowledge discovery. In: **Advances in evolutionary computing**. [s.l.] Springer, 2003. p. 819–845.

FREUND, Y.; SCHAPIRE, R. E. **A desicion-theoretic generalization of on-line learning and an application to boosting**. European conference on computational learning theory. **Anais**...Springer, 1995 Disponível em: <http://link.springer.com/chapter/10.1007/3-540-59119-2_166>

GACTO, M. J.; ALCALÁ, R.; HERRERA, F. Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures. **Information Sciences**, v. 181, n. 20, p. 4340–4360, 2011.

GOGUEN, J. A. The fuzzy Tychonoff theorem. **Journal of Mathematical Analysis and Applications**, v. 43, n. 3, p. 734–742, 1973.

GOLDBERG, D. E.; HOLLAND, J. H. Genetic algorithms and machine learning. **Machine learning**, v. 3, n. 2, p. 95–99, 1988.

GONÇALVES, L. B. et al. Inverted hierarchical neuro-fuzzy BSP system: a novel neuro-fuzzy model for pattern classification and rule extraction in databases. **IEEE Transactionson Systems, Man, and Cybernetics, Part C (Applications and Reviews)**, v. 36, n. 2, p. 236–248, 2006.

GRINGS, A. Regressão simbólica via programação genética: um estudo de caso com modelagem geofísica. 2006.

HASTIE, T. J.; TIBSHIRANI, R. J. **Generalized additive models**. [s.l.] CRC press, 1990. v. 43

HAYKIN, S. Neural networks, a comprehensive foundation. 1994.

HERRERA, F. Genetic fuzzy systems: taxonomy, current research trends and prospects. **Evolutionary Intelligence**, v. 1, n. 1, p. 27–46, 2008.

HERRERA, F.; MAGDALENA, L. Genetic fuzzy systems: A tutorial. **Tatra Mt. Math. Publ. (Slovakia)**, v. 13, p. 93–121, 1997.

HO, S.-Y. et al. Design of accurate classifiers with a compact fuzzy-rule base using an evolutionary scatter partition of feature space. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, v. 34, n. 2, p. 1031–1044, 2004.

HUANG, Z.; GEDEON, T. D.; NIKRAVESH, M. Pattern trees induction: A new machine learning method. **Fuzzy Systems, IEEE Transactionson**, v. 16, n. 4, p. 958–970, 2008.

HÜLLERMEIER, E. Fuzzy methods in machine learning and data mining: Status and prospects. **Fuzzy sets and Systems**, v. 156, n. 3, p. 387–406, 2005.

ISHIBUCHI, H.; NII, M. Fuzzy regression using a symmetric fuzzy coefficients and fuzzified neural networks. **Fuzzy Sets and Systems**, v. 119, n. 2, p. 273–290, 2001.

JANG, J.-S. ANFIS: adaptive-network-based fuzzy inference system. **IEEE transactions on systems, man, and cybernetics**, v. 23, n. 3, p. 665–685, 1993.

JANG, J.-S. R.; SUN, C.-T.; MIZUTANI, E. Neuro-fuzzy and soft computing; a computational approach to learning and machine intelligence. 1997.

JIAWEI, H.; KAMBER, M. Data mining: concepts and techniques. **San Francisco, CA, itd: Morgan Kaufmann**, v. 5, 2001.

KARR, C. Genetic algorithms for fuzzy controllers. **Ai Expert**, v. 6, n. 2, p. 26–33, 1991.

KEEL: A software tool to assess evolutionary algorithms for Data Mining problems (regression, classification, clustering, pattern mining and soon). Disponível em: <http://sci2s.ugr.es/keel/datasets.php>. Acesso em: 20 out. 2016.

KELLER, J. M.; YAGER, R. R.; TAHANI, H. Neural network implementation of fuzzy logic. **Fuzzy sets and systems**, v. 45, n. 1, p. 1–12, 1992.

KLEMENT, E. P.; MESIAR, R.; PAP, E. **Triangular norms**. [s.l.] Springer Science & Business Media, 2013. v. 8

KOSHIYAMA, A. S.; VELLASCO, M. M.; TANSCHEIT, R. UM CONTROLADOR FUZZY-GENÉTICO BASEADO EM PROGRAMAÇÃO GENÉTICA. 24 fev. 2016.

KOZA, J. R. **Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems**. [s.l.] Stanford University, Departmentof Computer Science, 1990.

KOZA, J. R. **Genetic programming: on the programming of computers by means of natural selection**. [s.l.] MIT press, 1992a.v. 1

KOZA, J. R. **Genetic programming: on the programming of computers by means of natural selection**. [s.l.] MIT press, 1992b. v. 1

LEGARDA, O.; VELLASCO, M.; TANSCHEIT, R. **RandomFIS: um Sistema de classificação Fuzzy para Problemas de Alta Dimensionalidade**. [s.l.] PUC-Rio, 2016.

LINDEN, R. **Algoritmos genéticos: uma importante ferramenta da inteligência computacional**. [s.l.] Brasport, 2006a.

LINDEN, R. **Algoritmos genéticos: uma importante ferramenta da inteligência computacional**. [s.l.] Brasport, 2006b.

LIU, Y. et al. A self-scaling instruction generator using cartesian genetic programming. In: **Genetic Programming**. [s.l.] Springer, 2011. p. 298–309.

MAIA, R. D. **Projeto de Estruturas Neuro-Fuzzy do Tipo Takagi-Sugeno Utilizando Programação Genética**. [s.l.] Universidade Federal de Minas Gerais, 2005.

MAMDANI, E. H. Applicationof fuzzy algorithms for control of simple dynamic plant. **Electrical Engineers, Proceedings of the Institution of**, v. 121, n. 12, p. 1585–1588, 1974.

MAMDANI, E. H.; ASSILIAN, S. An experiment in linguistic synthesis with a fuzzy logic controller. **International journal of man-machine studies**, v. 7, n. 1, p. 1–13, 1975.

MARQUES, L. G. Programação genética paralela com Pareto: uma ferramenta para modelagem via regressão simbólica. 2013.

MCKAY, B.; WILLIS, M. J.; BARTON, G. W. **Using a tree structured genetic algorithm to perform symbolic regression**. Genetic Algorithms in Engineering Systems: Innovations and Applications, 1995. GALESIA. First International Conference on (Conf. Publ. No. 414). **Anais**...IET, 1995 Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=501943>. Acesso em: 8 mar. 2016

MENDEL, J. M. Fuzzy logic systems for engineering: a tutorial. **Proceedings of the IEEE**, v. 83, n. 3, p. 345–377, 1995.

MENDEL, J. M. **Uncertain rule-based fuzzy systems: introduction and new directions**. [s.l.] Springer, 2017.

MERINO, J. S. P.; TANSCHEIT, R.; VELLASCO, M. **Síntese Automática de Sistemas de Inferência Fuzzy para Classificação**. [s.l.] PUC-Rio, 2015.

MICHALEWICZ, Z. **Genetic algorithms + data structures= evolution programs**. [s.l.] Springer Science & Business Media, 2013.

MILLER, J. F. **Cartesian genetic programming**. [s.l.] Springer, 2011.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. **Sistemas Inteligentes-Fundamentos e Aplicações**, v. 1, n. 1, 2003.

NAYAK, P. C. et al. A neuro-fuzzy computing technique for modeling hydrological time series. **Journal of Hydrology**, v. 291, n. 1, p. 52–66, 2004.

PEDRYCZ, W. Why triangular membership functions? **Fuzzy sets and Systems**, v. 64, n. 1, p. 21–30, 1994.

PEDRYCZ, W.; ROCHA, A. F. Fuzzy-set based models of neurons and knowledge-based networks. **IEEE Transactionson Fuzzy Systems**, v. 1, n. 4, p. 254–266, 1993.

POLAT, K.; GÜNEŞ, S. An expert system approach based on principal component analysis and adaptive neuro-fuzzy inference system to diagnosis of diabetes disease. **Digital SignalProcessing**, v. 17, n. 4, p. 702–710, 2007.

POLI, R. Parallel distributed genetic programming. **COGNITIVE SCIENCE RESEARCH PAPERS-UNIVERSITY OF BIRMINGHAM CSRP**, 1996.

POLI, R. et al. **A field guide to genetic programming**. [s.l.] Lulu. com, 2008.

REZENDE, S. O. **Sistemas inteligentes: fundamentos e aplicações**. [s.l.] Editora Manole Ltda, 2003.

ROSS, T. J. **Fuzzy logic with engineering applications**. [s.l.] John Wiley& Sons, 2009.

SANTOS, AA.; DO AMARAL, J. L. M. **Síntese de Árvores de padrões Fuzzy Através da Programação Genética Cartesiana**. [s.l: s.n.].

SCHWEFEL, H.-P. P. **Evolution and optimum seeking: the sixth generation**. [s.l.] John Wiley& Sons, Inc., 1993.

SCHWEIZER, B.; SKLAR, A. **Probabilistic metric spaces**. [s.l.] Courier Corporation, 2011.

SENGE, R.; HÜLLERMEIER, E. **Pattern trees for regression and fuzzy systems modeling**. Fuzzy Systems (FUZZ), 2010 IEEE International Conference on. **Anais** ...IEEE, 2010a Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp ?arnumber=5584231>. Acesso em: 26 jan. 2016

SENGE, R.; HÜLLERMEIER, E. **Pattern trees for regression and fuzzy systems modeling**. Fuzzy Systems (FUZZ), 2010 IEEE International Conference on. **Anais**...IEEE, 2010b Disponível em : <http://ieeexplore.ieee.org/xpls/abs_all.jsp? arnumber=5584231>. Acesso em: 26 jan. 2016

SENGE, R.; HULLERMEIER, E. Top-down induction of fuzzy pattern trees. **Fuzzy Systems, IEEE Transactionson**, v. 19, n. 2, p. 241–252, 2011.

SONG, Q.; CHISSOM, B. S. Fuzzy time series and its models. **Fuzzy sets and systems**, v. 54, n. 3, p. 269–277, 1993.

SOULE, T. **Code growth in genetic programming**. [s.l.] University of Idaho, 1998.

SOULE, T.; HECKENDORN, R. B. Ananalysis of the causes of code growth in genetic programming. **Genetic Programming and Evolvable Machines**, v. 3, n. 3, p. 283–309, 2002.

TAKAGI, T.; SUGENO, M. **Derivation of fuzzy control rules from human operator's control actions**. Proceedings of the IFAC symposium on fuzzy information, knowledge representation and decision analysis. **Anais**...sn, 1983

TAKAGI, T.; SUGENO, M. Fuzzy identification of systems and its applications to modeling and control. **IEEE transactionson systems, man, and cybernetics**, n. 1, p. 116–132, 1985.

TANSCHEIT, R. Sistemas fuzzy. **Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro**, 2004.

THRIFT, P. R. **Fuzzy Logic Synthesis with Genetic Algorithms.** ICGA. **Anais**...1991Disponível em: <http://www.academia.edu/download/11215308/1991-thrift-icga91.pdf>. Acesso em: 20 dez. 2016

TORRA, V. A review of the construction of hierarchical fuzzy systems. **International journal of intelligent systems**, v. 17, n. 5, p. 531–543, 2002.

UCI Machine Learning Repository: Data Sets. Disponível em: <https://archive.ics.uci.edu/ml/datasets.html>. Acesso em: 20 out. 2016.

USBERTI, F. L. SIMANFIS: Simplificação da Arquitetura Neuro-Fuzzy ANFIS. 15 fev. 2016.

VAPNIK, V. N.; VAPNIK, V. **Statistical learning theory**. [s.l.] Wiley New York, 1998. v. 1

WANG, L.-X.; MENDEL, J. M. Generating fuzzy rules by learning from examples. **IEEE Transactionson systems, man, and cybernetics**, v. 22, n. 6, p. 1414–1427, 1992a.

WANG, L.-X.; MENDEL, J. M. Fuzzy basis functions, universal approximation, and orthogonal least-squares learning. **IEEE transactionson Neural Networks**, v. 3, n. 5, p. 807–814, 1992b.

Weka 3 - Data Mining with Open Source Machine Learning Software in Java. Disponível em: <http://www.cs.waikato.ac.nz/ml/weka/datasets.html>. Acesso em: 24 jul. 2017.

WITTEN, I. H.; FRANK, E. **Data Mining: Practical machine learning tools and techniques**. [s.l.] Morgan Kaufmann, 2005.

YEGNANARAYANA, B. **Artificial neural networks**. [s.l.] PHI Learning Pvt. Ltd., 2009.

ZADEH, L. A. Fuzzy sets. **Information and control**, v. 8, n. 3, p. 338–353, 1965.

ZHANG, W. **Complete any time beam search**. AAAI/IAAI. **Anais**...1998 Disponível em: <http://www.aaai.org/Papers/AAAI/1998/AAAI98-060.pdf>. Acesso em: 18 jan. 2017

ZIMMERMANN, H.-J. Fuzzy Control. In: **Fuzzy Set Theory—and Its Applications**. [s.l.] Springer, 1996. p. 203–240.

ZOUNEMAT-KERMANI, M.; TESHNEHLAB, M. Using adaptive neuro-fuzzy inference system for hydrological time series prediction. **Applied Soft Computing**, v. 8, n. 2, p. 928–936, 2008.