UM FRAMEWORK DE FUZIFICADOR ORIENTADO A OBJETOS EM JAVA

Aislan Fernandes Pereira
Centro Federal de Educação Tecnológica da Paraíba
E-mail: aislanfp@bol.com.br

Resumo

O presente trabalho apresenta um framework orientado a objetos implementado em Java de fuzificador. Esse framework pode ser multifenômeno e cada uma de suas variáveis lingüísticas pode ser representada por mais de uma amostra. A função de pertinência de cada variável lingüística pode ser implementada por diferentes cálculos numéricos de interpolação. Esse framework representa, dentro de uma arquitetura MVC, formada das camadas de apresentação, lógica da aplicação e serviços, a camada lógica da aplicação.

Palavras-chave: Lógica Nebulosa. Inteligência Artificial. Conjuntos Difusos. Framework.

1. Introdução

O campo da Inteligência Artificial contém diversas técnicas de implementação e dentre as mais recentes encontramos a Lógica Fuzzy, também conhecida como Conjuntos Difusos ou Lógica Nebulosa, que consiste em aproximar a decisão computacional da decisão humana. Quer dizer, a decisão de uma máquina não se resume apenas a um "sim" ou um "não" mas também têm decisões "abstratas", do tipo "um pouco menos", "talvez não", e outras mais, também, chamadas de variáveis lingüísticas, que representam as decisões humanas.

O uso de conjuntos nebulosos possibilita a representação dessas decisões. Uma função de pertinência associa um número real no intervalo [0,1] para cada elemento pertencente a esse conjunto nebuloso. Esse número representa o grau de possibilidade ou pertinência de que o esse elemento venha a pertencer a esse conjunto.

A entrada de um sistema fuzzy ou mesmos outros sistemas que assim utilizem valores numéricos que representem um determinado objeto não-mensurável é formado por um módulo chamado de fuzificador. Esse fuzificador transforma os valores precisos como por exemplo o valor 40° de um fenômeno como a temperatura em graus de pertinência como 0.2, 0.8 e 0.2 relacionados a cada uma de suas respectivas variáveis lingüísticas que poderiam ser o conjunto das temperaturas altas, o conjunto das temperaturas mornas e o conjunto das temperaturas baixas. Esse cálculo pode ser feito através de uma função de pertinência que na maioria das aplicações práticas são do tipo gaussiana, triangulares, trapezoidais, crescentes ou decrescentes.

Normalmente, um fuzificador é implementado sem qualquer portabilidade ou flexibilidade devido a uma construção proprietária ao seu sistema fuzzy.

Um framework (arcabouço) é um conjunto de classes cooperantes que constroem um projeto reutilizável para uma específica classe de software, que facilita a construção de soluções específicas, como por exemplo, fuzificadores, bastando, apenas, o desenvolvedor se concentrar em questões específicas de implementação ao utilizar o framework.

A linguagem Java permite ao desenvolvedor construir aplicações orientados a objetos,. que podem representar casos do mundo real, mas, necessariamente, nem todos os objetos encontrados numa solução modelam esses casos. Um fuzificador como sendo uma caso do mundo real pode ser modelado em objetos.

2. Objetivos

- **Geral:**

  Ser reutilizável na implementação de qualquer módulo de entrada de um sistema fuzzy.

- **Específicos:**

1. Adicionar um ou mais fenômenos com seus respectivos valores lingüísticos e amostras desses valores;
2. Gerar os graus de pertinência para cada valor de entrada de um determinado fenômeno;
3. Fornecer suporte para adicionar um interpolador a ser utilizado por um valor lingüístico;
4. Fornecer suporte para adicionar um cálculo numérico a ser utilizado na geração da equação que descreve o comportamento de uma amostra;
5. Redefinir uma amostra através da realimentação de um ou mais pontos da mesma.

3. Metodologia de Implementação

O framework é formado basicamente de quatro classes e uma interface de acordo com o paradigma orientado a objetos, chamados respectivamente de Fuzificador, Fenomeno, Variavel, Amostra e Interpolador. Uma interface adicional foi colocada com a finalidade de gerar os coeficientes da equação da função, que descreve a amostra de uma variável lingüística, chamado de SistemaLinear.

A classe Fuzificador foi criada com a responsabilidade de prover a entrada e a saída de dados, isto é, a entrada de valores precisos de um fenômeno, a saída de graus de pertinência, e também, com a capacidade de gerenciar um ou mais fenômenos.

A classe Fenomeno foi criada como uma representação do fenômeno de um sistema fuzzy e com a responsabilidade de gerenciar um ou mais variáveis lingüísticas desse fenômeno.

A classe Variavel foi concebida, também, como uma representação de uma variável lingüística, contendo uma função de pertinência responsável por gerar o grau de pertinência de um valor do fenômeno e podendo gerenciar um ou mais amostras que descrevem seu comportamento. A habilidade de adicionar uma ou mais amostras foi necessário porque em um intervalo de seus valores poderia acontecer de haver por exemplo o comportamento de uma função polinomial de primeiro grau e em outro intervalo o de uma função polinomial de segundo grau, logo para cada intervalo haveria uma amostra diferente.

A classe Amostra foi criada com a finalidade de conter os valores simulando um comportamento em um determinado intervalo de uma variável lingüística. Esses valores são o valor preciso de um fenômeno e seu grau de pertinência já conhecidos, também chamados de coordenadas x e y dessa classe. Essa classe contém a capacidade de realimentação ("aprender"); quando adicionamos uma ou mais novas coordenadas, tornando o comportamento ainda mais preciso. Como opcional, essa classe pode ainda gerar a equação da função que descreve seu comportamento, através do uso do cálculo numérico implementado na interface SistemaLinear.

A interface Interpolador foi criada, a fim de representar a implementação de uma função de pertinência de uma variável lingüística. A interpolação é um cálculo numérico, que

disposto de alguns pontos já conhecidos, abscissa x e coordenada y, podemos encontrar um valor desconhecido de y para um novo valor de x através de uma função aproximada desses pontos conhecidos. Essa interface possibilita também a uma variável lingüística mudar o cálculo numérico de sua função de pertinência, podendo ser do tipo polinomial, exponencial, logarítmica ou trigonométrica, desde de que uma matriz de pontos de uma amostra e um valor do fenômeno sejam seus parâmetros de entrada para o processamento.

Podemos verificar que a classe Fuzificador contém uma ou mais classes Fenomeno, e essa por sua vez é formada por uma ou mais classes Variavel, que contém pelo menos uma classe Amostra. E cada classe Variavel chama o método de interpolação da interface Interpolador para executar a sua função de pertinência. Esses relacionamentos estão ilustrados na Figura 2 onde temos um exemplo de um fuzificador com apenas um fenômeno, três variáveis lingüísticas com cada uma dessa variável com uma amostra diferente e utilizando o mesmo interpolador do tipo polinomial que usa a interpolação de Lagrange.

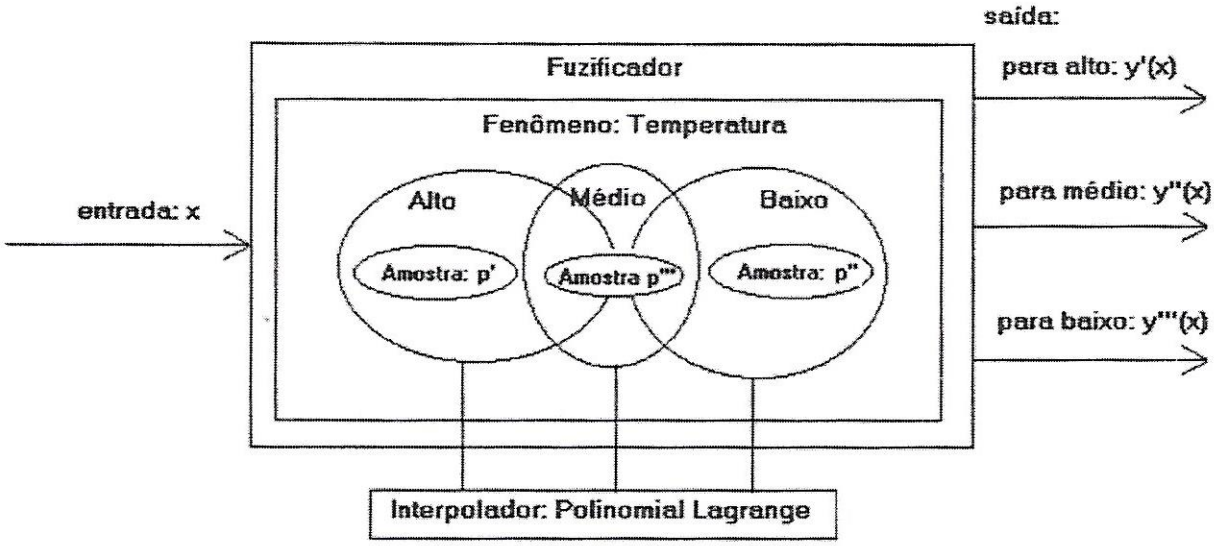

FIGURA 1: Exemplo de um caso de um fuzificador

4. Plataforma de hardware e software

Na escolha do software e do hardware necessários para a implementação e funcionamento do framework foram considerados três aspectos importantes: interoperabilidade, orientado a objetos e flexibilidade. A primeira é importante para não haver limitação na escolha do sistema operacional onde será executado o fuzificador. A segunda é necessária para refletir a realidade de um fuzificador formado de um conjunto de entidades interrelacionadas. E o terceiro é essencial para definirmos o nosso fuzificador como genérico.

Na escolha do software, todos os aspectos antes descritos são preenchidos pela linguagem de programação Java 2 Platform, Standard Edition, version 1.3. E a ferramenta computacional escolhida para o desenvolvimento do fuzificador utilizando essa plataforma foi o Kawa versão 3.21.

Na escolha do hardware recaiu uma configuração mínima de um microcomputador compatível com PC com um processador 486 de 50 MHZ, 16 MB de memória de RAM, monitor de vídeo no padrão VGA.

O fuzificador foi desenvolvido utilizando a estrutura física existente nos laboratórios de redes de computadores do CEFET-PB, utilizando um microcomputador Pentium II Celeron de 300 MHz, com memória RAM de 64 Mb, disco rígido de 6,3 Gb, monitor SVGA de 15 polegadas com tela plana e resolução 800 X 600 dpi high color, unidade de disco flexível de

1.44Mb e unidade de CD-ROM 40X.

5. Simulações e resultados

Os resultados das simulações foram gerados com base no método de interpolação de Lagrange, necessário para a interface Interpolador, e o método de eliminação de Gauss para a interface SistemaLinear. Havia uma restrição no algoritmo do método de Gauss pois não poderia receber o valor de uma abscissa como 0 (zero).

Em nossos testes definimos um fuzificador contendo um fenômeno, com duas variáveis lingüísticas, *A* e *B*, e cada uma contendo uma amostra. A amostra de *A* continha uma matriz de três pontos e a amostra de *B* continha uma matriz de dois pontos, como mostradas pela Tabela 1 e a Tabela 2. A primeira coluna, índice 0 (zero), dessas matrizes representam as abscissas.

Os resultados obtidos, grau de pertinência de cada variável com base em cada valor de um fenômeno, estão mostrados na Tabela 3.

TABELA 1: Matriz de pontos da amostra *A*

[0][0]	1
[1][0]	2
[2][0]	3
[0][1]	3
[1][1]	7
[2][1]	13

TABELA 2: Matriz de pontos da amostra *B*

[0][0]	9
[1][0]	10
[0][1]	18
[1][1]	20

TABELA 3: Resultado dos testes

Valor do Fenômeno	Grau de Pertinência (0,1)	
	Variável *A*	Variável *B*
5.0	0.129166666666666668	0.0
9.0	0.37916666666666665	0.75
14.0	0.8791666666666667	1.1666666666666667

Foi levado em conta que a amostra de *A* continha o valor mínimo, do fenômeno de 0 e o valor máximo de 15, e a amostra *B* com o valor mínimo de 8 e o valor máximo de 20. E esses valores de mínimo e máximo mostram que em um intervalo de valores do fenômeno podemos ter a partir de, uma valor do fenômeno (entre 8 e 15) graus de pertinência diferentes de 0 (zero) em relação à variável *A* e *B* ao mesmo tempo. As equações obtidas pelas amostra estão ilustradas na Tabela 4.

TABELA 4: Equações das amostras

Amostra	Equação
Da variável *A*	$y(x) = x^2 + x + 1.0$
Da variável *B*	$y(x) = 2x$

6. Considerações finais e perspectivas futuras

O uso de um framework traz grandes contribuições para desenvolvedores de sistemas fuzzy, porque permitem uma implementação mais concentrada em questões específicas de fuzificadores, isto é, sem haver uma certa preocupação em implementar toda uma estrutura de fuzificador.

No presente trabalho, foram levados em conta, apenas, recursos de cálculos numéricos de interpolação, isto é, não se consideraram extrapolações. É importante lembrar que interpolações apresentam erros de arredondamento e erro de truncamento, portanto, uma análise prévia deve ser feita para verificar se há a necessidade de tratamento de tais erros.

A classe Amostra foi concebida para trabalhar com funções polinomiais, porque tais funções são normalmente utilizadas nos casos mais comuns para resolução de problemas. A adição da interface SistemaLinear para implementar a geração da equação do comportamento de uma Amostra, também, está ligado a problemas relacionados com funções polinomiais.

Um conhecimento de um sistema fuzzy real não foi levado em conta na validação da definição do framework citado neste documento.

Em um trabalho futuro poderia haver uma implementação da camada de apresentação, onde seriam definidas as interfaces (usuário/sistema) de tratamento do framework, seria um ótimo recurso para facilitar o uso do framework e se poderia investigar implementações das propriedades de união, intercessão, potenciação e radiciação do fuzificador.

7. Referências bibliográficas

LARMAN, C. **Utilizando UML e padrões**. Porto Alegre, 2000.

DEITEL, H. M.. **Java: como programar**. Ed. Bookman, 2001.

GAMMA, E.; et al. **Padrões de projeto**: soluções reutilizáveis de software orientado a objetos. Porto Alegre: Bookman, 2000.

ALTROCK, C. V. **Fuzzy logic & neurofuzzy applications explained**. Prentice-Hall PTR, 1995.

LIN, C.; LEE, C.S. G. **Neural fuzzy systems: a neuro-fuzzy synergism to intelligent systems**. Prentice-Hall PTR, 1996.

MENDEL, J M. **Fuzzy logic systems for engineering: a tutorial**. Proc. of the IEEE, 1995. v. 83, No. 3.

DRIANKOV, D.; et al. **An introduction to fuzzy control**. Springer-Verlag, 1993.