Atividade Lógica Fuzzy Aplicada

20209000617 Francisco do Carmo Amorim – Inteligência Computacional Aplicada - 2023.2

1) INTRODUÇÃO

A lógica fuzzy é capaz de modelar situações através de um processo de incerteza e imprecisão de conjuntos similar à tomada de decisão humana. Para tanto, são executadas as seguintes etapas:

- Fuzzificação: Mapeamento das variáveis de entrada em valores fuzzy pela associação de um valor de entrada ao valor de saída da função de pertinência (normalmente os valores de saída são normalizados).

- Base de Regras: Definições usando uma base de conhecimento sobre o problema para relacionar valores fuzzy.

- Inferência: Relações **Se-Entao** entre conjuntos fuzzy;

- Desfuzzificação: Mapeia os valores fuzzy de volta em uma saída numérica para o problema.

As situações propostas são duas, portanto dividiremos este resumo em duas partes, cada qual com a descrição da situação envolvida, a estruturação do sistema fuzzy, o código e os resultados obtidos.

2.1) Avaliação do primeiro problema:

"**Especifique e projete um sistema Fuzzy de avaliação de crédito (Credit Score) que leve em consideração a renda (0-100), a dívida (0-100) e o patrimônio do cliente (0-100), de forma que o valor do Score esteja entre 0 e 1.000.**

Mostre as funções de pertinência, inferência e especifique a desfuzzificação. Defina três exemplos e mostre os resultados."

Devemos ter em mente que no momento de definir a fuzzificação, quanto mais classes criarmos, mais combinações existirão, portanto, para um número fixo de exemplos, a representatividade do espaço total será menor.

Começamos definindo como segue para cada categoria:

- Renda: [Baixa, Média, Alta];
- Dívida: [Baixa, Alta];
- Patrimônio: [Baixo, Médio, Alto].

Com isso temos um total de $3 \times 2 \times 3 = 18$ possibilidades, partimos agora para a escolha das funções de pertinência de entrada e as definições matemáticas (intervalos) para cada classificação.

Para a renda iremos adotar a curva gaussiana, por prover uma definição maior de pertencimento da classe central para valores próximos do pico, e uma transição suave para outras classes. Nesse sentido, lembramos que a função gaussiana é dada pela fórmula:

$$\mu_R(x) = e^{\frac{-(x-c)^2}{2\cdot\sigma^2}}$$

De modo que selecionando o centro 'c' como $(c_1; c_2; c_3) = (0; 50; 100)$ para cada classe e um desvio padrão 'σ' de $(\sigma_1; \sigma_2; \sigma_3) = (15; 15; 15)$, obtemos a seguinte figura:

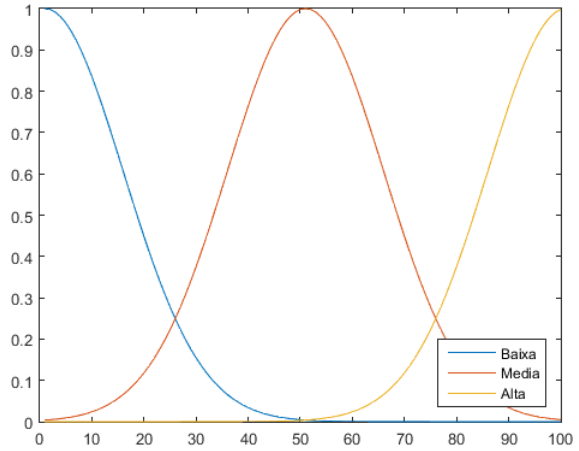

Figura 1 - Função de pertinência gaussiana para as três classes de renda.

Para a dívida, adotamos a função trapezoidal por possuir um núcleo verdade, isto é, para um dado intervalo de x, temos que a saída é 1 – isso reduz a incerteza do conjunto, lembrando que a função trapezoidal pode ser descrita como:

$$\mu_v(x) = \begin{cases} 0, & x < v_1 \\ \frac{x-v_1}{v_2-v_1}, & v_1 \leq x \leq v_2 \\ 1, & v_2 \leq x \leq v_3 \\ \frac{v_4-x}{v_4-v_3}, & v_3 \leq x \leq v_4 \\ 0, & x > v_4 \end{cases}$$

Tomando duas funções trapezoidais temos então oito parâmetros para configurar, definimos:

$min\_low\_debt = 0$
$max\_low\_debt = 10$
$min\_high\_debt = 60$
$max\_high\_debt = 100$

Como sendo indicativos de menor/maior valor para baixa dívida e menor/maior valor para alta dívida respectivamente. E então tomamos o vetor v de 8 parâmetros como:

$$v = (v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8)$$

$v_1 = 0$
$v_2 = min\_low\_debt$
$v_3 = max\_low\_debt$
$v_4 = 50$

$v_5 = 0$
$v_6 = min\_high\_debt$
$v_7 = max\_high\_debt$
$v_8 = 100$

O figura 2 representa a saída para esse conjunto com a função de pertinência trapezoidal e os parâmetros definidos acima.

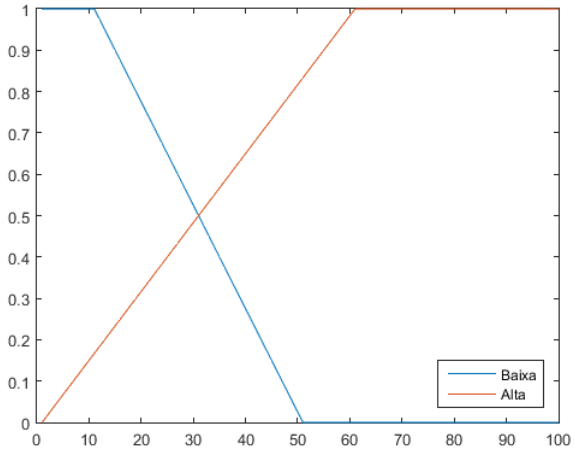

Figura 2 - Função de pertinência trapezoidal para as duas classes de dívida.

Por fim, escolhemos a função gaussiana novamente para a categoria de patrimônio, usaremos os mesmos parâmetros: Centro 'c' como 0;50;100 para cada classe e um desvio padrão 'σ' de 15. A figura 3 mostra o resultado obtido.

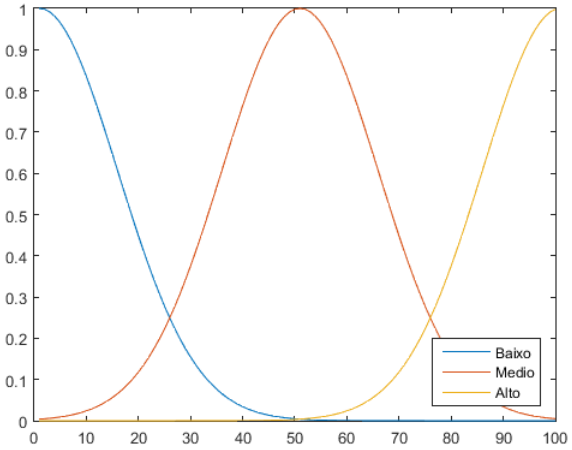

Figura 3 - Função de pertinência gaussiana para as três classes de patrimônio.

Definimos agora as funções de pertinência da saída, para uma saída classificatória de 0 a 1000 (score), usaremos a função de pertencimento triangular pelo seu crescimento gradual e de incremento constante, desse modo, seja a função dada por:

$$\mu_A(x) = \begin{cases} 0 & if\ x \le a \\ \dfrac{x-a}{b-a} & if\ a \le x \le b \\ \dfrac{c-x}{c-b} & if\ b \le x \le c \\ 0 & if\ x \ge c \end{cases}$$

A partir da descrição anterior, precisamos determinar nove parâmetros relativos a cada uma das três funções triangulares, para tanto os seguintes valores foram adotados:

$$(a_1; b_1; c_1) = (0; 0; 35)$$
$$(a_2; b_2; c_2) = (20; 50; 80)$$
$$(a_3; b_3; c_3) = (65; 100; 100)$$

O gráfico de saída é mostrado na figura 4 a seguir:

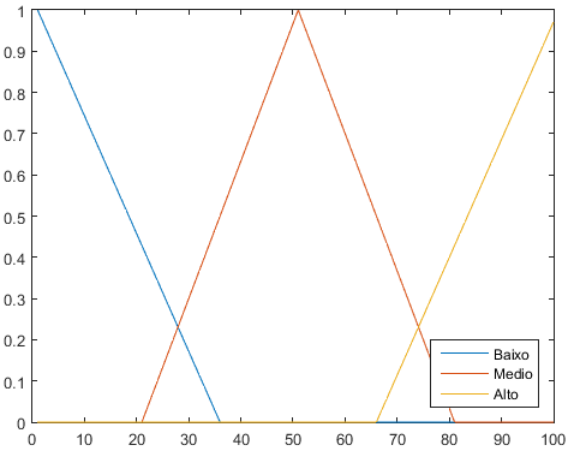

Figura 4 - Função de pertinência triangular para as três classes de crédito.

Compete-nos agora definir as regras para a inferência, de modo que as seguintes são tomadas:

(1) Se Renda Baixa E Patrimônio Baixo, então crédito baixo.
(2) Se Renda Baixa E Patrimônio Médio, então crédito médio.
(3) Se Renda Baixa E Patrimônio Alto, então crédito alto.
(4) Se Renda Alta E Dívida Baixa, então crédito alto.
(5) Se Renda Alta E Dívida Alta, então crédito baixo.
(6) Se Renda Alta E Patrimônio Alto, então crédito alto.
(7) Se Renda Alta E Patrimônio Médio, então crédito alto.
(8) Se Dívida Alta E Patrimônio Baixo, então crédito baixo.
(9) Se Dívida Baixa E Patrimônio Médio, então crédito médio.

Após o processo de inferência, devemos agora nos ater à desfuzzificação, para tanto iremos somar os resultados de cada classificação (baixo, médio e alto), e em seguida iremos obter a saída, para a qual escolhemos o método do Centro de Área. Dado que inicialmente o sistema estava configurado em uma escala [0, 100], o valor final foi multiplicado por 10 para obter a escala de [0, 1000]. Os resultados obtidos para cada exemplo são mostrados nas figuras a seguir:

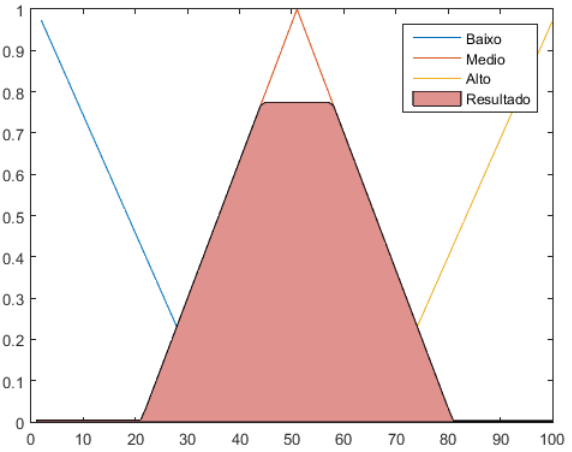

Figura 5 – Avaliação de crédito pelo algoritmo para $(r, d, p) = (40, 20, 50)$. O score obtido para este exemplo foi de 499,99.

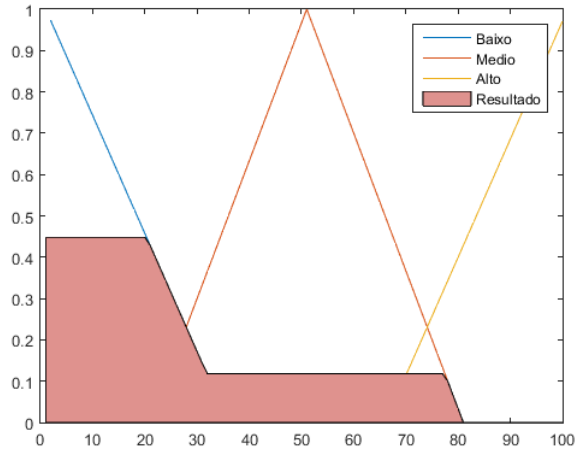

Figura 6 – Avaliação de crédito pelo algoritmo para $(r, d, p) = (10, 10, 20)$. O score obtido para este exemplo foi de 170,60.

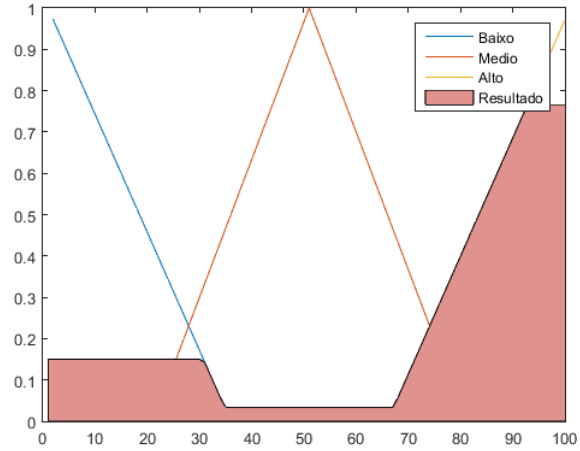

Figura 7 – Avaliação de crédito pelo algoritmo para $(r, d, p) = (90, 10, 90)$. O score obtido para este exemplo foi de 855,28.

Com os exemplos anteriores foi possível perceber os três casos descritos pela saída (score baixo, médio e alto), cabe porém ressaltar que as regras de base apresentam caráter forte no que concerne à sensibilidade na saída, isso pode ser visto na figura 7, onde esperávamos mais parcela na região central do que na região inicial. Portanto, a adição de mais regras que expanda o domínio de conhecimento do sistema pode diminuir a dispersão e promover maior robustez à saída.

2.2) Código referente ao primeiro problema

O código para execução dos passos anteriormente descritos foi executado no MATLAB R2015a (MathWorks) e é mostrado a seguir:

```
%% Problema da avaliacao de credito

function main()
%valores de entrada
x=0:100;

%ex = [renda, divida, patrimonio]
ex1 = [40, 20, 50];
ex2 = [10, 10, 20];
ex3 = [90, 10, 90];

%para testar para outros exemplos basta
mudar ex1() para ex2() ou ex3()

r = ex3(1);
d = ex3(2);
p = ex3(3);

%A palette de cor foi tomada de Hex para
RGB e entao feita scaling para
%ser lida pelo MatLab (dividimos o valor
por 255)
palette = [223/255 146/255 142/255]

% Funções de Pertinência da Entrada
%usando parametros de renda

sigma=15;
renda_baixa = GaussMF(x, sigma, 0);
renda_media = GaussMF(x, sigma, 50);
renda_alta  = GaussMF(x, sigma, 100);

figure
plot(renda_baixa) ;
hold on;
plot(renda_media);
plot(renda_alta);
axis([0 100 0 1])
legend('Baixa', 'Media', 'Alta',
'Location', 'Southeast')
hold off;


%usando parametros de divida

min_low_debt = 0;
max_low_debt = 10;
min_high_debt = 60;
max_high_debt = 100;
divida_baixa = TrapezoidalMF(x, 0,
min_low_debt, max_low_debt, 50);
divida_alta  = TrapezoidalMF(x, 0,
min_high_debt, max_high_debt, 100);

figure
plot(divida_baixa);
hold on;
plot(divida_alta);
axis([0 100 0 1]);
legend('Baixa', 'Alta', 'Location',
'Southeast');
hold off;

%usando parametros de patrimonio

sigma=15;
patr_baixo = GaussMF(x, sigma, 0);
patr_medio = GaussMF(x, sigma, 50);
patr_alto  = GaussMF(x, sigma, 100);

figure
plot(patr_baixo) ;
hold on;
plot(patr_medio);
plot(patr_alto);
axis([0 100 0 1]);
```

```
legend('Baixo', 'Medio', 'Alto',
'Location', 'Southeast');
hold off;


% Funções de Pertinência da Saída

cred_baixo = TriangularMF(x, 0, 0, 35);
cred_medio = TriangularMF(x, 20, 50, 80);
cred_alto  = TriangularMF(x, 65, 100,
100);

figure
plot(cred_baixo);
hold on;
plot(cred_medio);
hold on;
plot(cred_alto);
axis([0 100 0 1]);
legend('Baixo', 'Medio', 'Alto',
'Location', 'Southeast');

%Regras

regra_1 =
min(renda_baixa(r),patr_baixo(p));
uc1 = min(regra_1,cred_baixo);

regra_2 =
min(renda_baixa(r),patr_medio(p));
uc2 = min(regra_2,cred_medio);

regra_3 =
min(renda_baixa(r),patr_alto(p));
uc3 = min(regra_3,cred_alto);

regra_4 =
min(renda_alta(r),divida_baixa(d));
uc4 = min(regra_4,cred_alto);

regra_5 =
min(renda_alta(r),divida_alta(d));
uc5 = min(regra_5,cred_baixo);

regra_6 = min(renda_alta(r),patr_alto(p));
uc6 = min(regra_6,cred_alto);

regra_7 =
min(renda_alta(r),patr_medio(p));
uc7 = min(regra_7,cred_alto);

regra_8 =
min(divida_alta(d),patr_baixo(p));
uc8 = min(regra_8,cred_baixo);

regra_9 =
min(divida_baixa(d),patr_medio(p));
uc9 = min(regra_9,cred_medio);

figure
plot(uc1);
hold on;
plot(uc2);plot(uc3);plot(uc4);plot(uc5);pl
ot(uc6);
plot(uc7);plot(uc8);plot(uc9);

hold off;

%Desfuzzificação

Y=max([uc1 ;uc2 ;uc3 ;uc4 ;uc5; uc6 ;uc7;
uc8 ;uc9]);
figure

plot(cred_baixo);hold on;
plot(cred_medio); plot(cred_alto);
area(Y, 'FaceColor', palette);
axis([0 100 0 1]);
legend('Baixo', 'Medio', 'Alto',
'Resultado','Location', 'Northeast');

% Saída
% Calculo utilizando o método do centro de
area
w = 10*(Y.*x)/Y

end



%% Funcoes de pertinencia implementadas


function tri = TriangularMF(x, a, b, c)

    for i = 1:numel(x)
        if x(i) < a
            tri(i) = 0;
        elseif a <= x(i) && x(i) <= b
            tri(i) = (x(i) - a) / (b - a);
        elseif b < x(i) && x(i) <= c
            tri(i) = 1 - (x(i) - b) / (c -
b);
        elseif x(i) > c
            tri(i) = 0;
        end
    end
end


function gauss = GaussMF(x, sigma, m)
    for i=1:numel(x)
        gauss(i)=exp(-1/2*(1/sigma*(x(i)-
m))^2);
    end
end


function trap = TrapezoidalMF(x,a,b,c,d)
    for  i= 1:numel(x)
        if x(i) < a
            trap(i) = 0;
```

```
        elseif a <= x(i) && x(i) <= b
            trap(i) = (x(i) - a) / (b -
a);
        elseif b < x(i) && x(i) <= c
            trap(i) = 1
        elseif c < x(i) && x(i) <= d
            trap(i) = 1 - (x(i) - c) / (d-
c);
        elseif d < x(i)
            trap(i)=0;
        end
    end
end
```

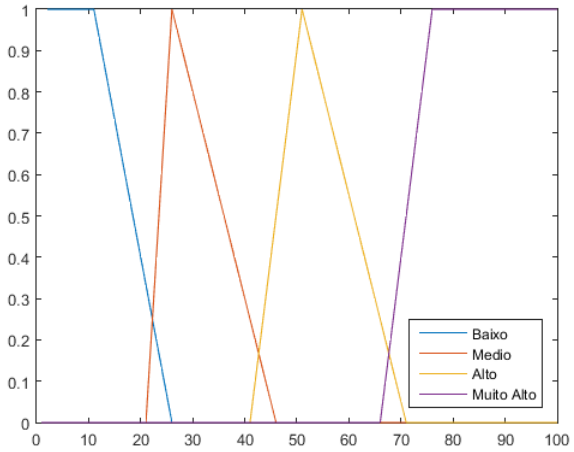

Figura 8 - Função de pertinência triangular e trapezoidal número de carros.

2.3) Avaliação do segundo problema:

"**Especifique e projete um sistema Fuzzy de gerenciamento de um conjunto de semáforos em um cruzamento composto de 2 estágios. Mostre as funções de pertinência, universo de discurso, inferência e especifique a desfuzzificação.**
Defina três exemplos e mostre os resultados."

Para a situação dada temos 4 sensores T_1, T_2, T_3 e T_4, onde cada um registra o número de carros passando pela rua naquele momento, portanto, nosso sistema fuzzy será alimentado com estes dados de entrada para fornecer um tempo de passagem (sinal verde). Outro ponto a ser mencionado é que para uma dada fase podemos ter o maior tempo possível ou no sinal verde ou no sinal vermelho, contudo, todas as três luzes devem ocorrer durante uma fase para cada par de sinal (cruzamentos indevidos não são permitidos).
Diante do exposto, fica clara que na saída devemos ter uma indicação do tempo de saída do espaço nebuloso para então traduzirmos para um valor numérico correspondente ao problema real. A situação será descrita pelos seguintes parâmetros:

$$trafego\_baixo = 10$$
$$trafego\_medio = 25$$
$$trafego\_alto = 50$$
$$trafego\_muito\_alto = 75$$

Iremos abreviar as variáveis acima como BA, ME, AL e MA respectivamente. Esses serão os parâmetros base para as funções de pertinência triangular e trapezoidal na entrada. Definindo duas funções triangulares e duas trapezoidais temos portanto 14 parâmetros, tomaremos os seguintes valores:

$$(a_1; b_1; c_1) = (0; 0; trafego\_baixo\ ; 25)$$
$$(a_2; b_2; c_2) = (20; trafego\_medio; 45)$$
$$(a_3; b_3; c_3) = (40; trafego\_alto\ ; 70)$$
$$(a_4; b_4; c_4; d_4) = (65; trafego\_muito\_alto; 100; 100)$$

O resultado da configuração anterior é mostrado na figura 8 a seguir:

Para a saída iremos tomar a função gaussiana por possuir variações suavizadas ter uma classe mediana controlável pelo parâmetro do desvio padrão, de forma que definimos:

$$(c_1; \sigma_1) = (0; 10)$$
$$(c_2; \sigma_2) = (35; 10)$$
$$(c_3; \sigma_3) = (65; 10)$$
$$(c_4; \sigma_4) = (100; 10)$$

Elas irão representar as variáveis de tempo curto, médio, longo e muito longo em sequência. Para efeitos de simplificação, iremos também abrevia-las da seguinte forma: TC, TM, TL e TML na mesma sequência em que anteriormente foram mencionadas. O gráfico das formas definidas anteriormente é mostrado na figura 9:

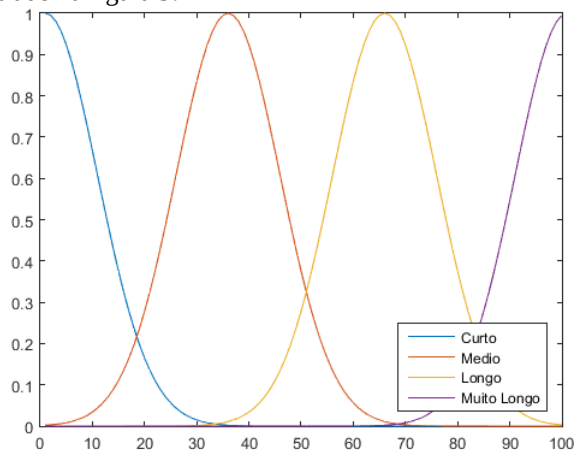

Figura 9 - Função de pertinência gaussiana para tempo de semáforo.

Em resumo, o universo de discurso será composto tanto pela entrada com as variáveis de $trafego\_baixo$, $trafego\_medio$, $trafego\_alto$ e $trafego\_muito\_alto$ como também pelas variáveis do conjunto de saída sendo elas tempo curto, médio, longo e muito longo. Partimos para definir então as regras de base para inferência do sistema fuzzy:

Sensores				Semáforos	
T_1	T_2	T_3	T_4	S_1/S_3	S_2/S_4
BA	BA	BA	BA	TC	TC
BA	BA	ME	AL	TM	TL
BA	ME	ME	AL	TM	TL
BA	ME	AL	BA	TL	TM
ME	AL	AL	AL	TL	TL
ME	BA	BA	ME	TM	TM
ME	ME	MA	BA	TML	TM
ME	AL	BA	MA	TL	TML

AL	ME	AL	ME	TL	TM
AL	AL	ME	ME	TL	TL
AL	BA	BA	AL	TL	TL
AL	ME	BA	BA	TL	TM
MA	MA	BA	MA	TML	TML
MA	BA	ME	BA	TML	TC
MA	ME	MA	AL	TML	TL
MA	BA	AL	BA	TML	TC
ME	MA	BA	AL	TM	TML
BA	ME	MA	ME	TML	TM
BA	AL	BA	BA	TC	TL

Com isso, podemos ver que mesmo em uma tabela de tamanho considerável, ainda estamos distantes da quantidade de possibilidades possíveis ($4 \times 4 \times 4 \times 4 = 256$), isso mostra que a escolha de elementos representativos em um espaço pequeno é difícil e exige a avaliação à priori da representatividade de cada possibilidade de entrada para uma dada saída. Para efeitos de código foram necessárias duas escritas, uma para regras tendo como saída S_1/S_3 representados por $\mu_{C1} - \mu_{C19}$ e área dada por Y_1 e centro de massa por w_1, e outra tendo como saída S_2/S_4 representados por $\mu_{C19} - \mu_{C38}$ e área dada por Y_2 e centro de massa por w_2.

Para o processo de desfuzzificação, tomaremos como método de saída o centro de massa da área da figura, esta é encontrada levando-se em conta todas as possibilidades de saída computadas pelo algoritmo para uma entrada.

Os exemplos adotados são ex1 = [1 30 15 60]; ex2 = [44 20 21 10];ex3 = [70 84 10 36]. Estes podem ser usados para compor a saída apenas mudando o parâmetro de T_1, T_2, T_3 e T_4 onde há ex1, pode-se colocar ex2 ou ex3.

2.4) Código referente ao segundo problema

O código para execução dos passos anteriormente descritos foi executado no MATLAB R2015a (MathWorks) e é mostrado a seguir:

```
%%Problema do semaforo de dois tempos


function main()
x = 0:100;

%A palette de cor foi tomada de Hex para
RGB e entao feita scaling para
%ser lida pelo MatLab (dividimos o valor
por 255)
palette = [223/255 146/255 142/255];

% Pares (1,3) & (2,4)
%ex = [T1 T2 T3 T4]
ex1 = [1 30 15 60];
ex2 = [44 20 21 10];
ex3 = [70 84 10 36];

T1 = ex1(1);
T2 = ex1(2);
T3 = ex1(3);
T4 = ex1(4);

% Funcoes de pertinencia da entrada

trafego_baixo =
TrapezoidalMF(x,0,0,10,25);
trafego_medio = TriangularMF(x,20,25,45);
trafego_alto = TriangularMF(x,40,50,70);
trafego_muito_alto =
TrapezoidalMF(x,65,75,100,100);

%figure
plot(trafego_baixo);hold on;
plot(trafego_medio); plot(trafego_alto);
plot(trafego_muito_alto);
axis([0 100 0 1]);
legend('Baixo', 'Medio', 'Alto','Muito
Alto', 'Location', 'Southeast')
hold off;

% Funcoes de pertinencia de saida

sigma = 10;
tempo_curto = GaussMF(x,sigma,0);
tempo_medio = GaussMF(x,sigma,35);
tempo_longo = GaussMF(x,sigma,65);
tempo_muito_longo = GaussMF(x,sigma,100);

%figure
plot(tempo_curto); hold on;
plot(tempo_medio); plot(tempo_longo);
plot(tempo_muito_longo);
axis([0 100 0 1]);
legend('Curto', 'Medio', 'Longo','Muito
Longo', 'Location', 'Southeast');
hold off;

% Base de regras & Inferencia

uc1 =
min(min(min(min(trafego_baixo(T1),trafego_
baixo(T2)), trafego_baixo(T3)),
trafego_baixo(T4)), tempo_curto);
uc2 =
min(min(min(min(trafego_baixo(T1),trafego_
baixo(T2)), trafego_medio(T3)),
trafego_alto(T4)), tempo_medio);
uc3 =
min(min(min(min(trafego_baixo(T1),trafego_
medio(T2)), trafego_medio(T3)),
trafego_alto(T4)), tempo_medio);
uc4 =
min(min(min(min(trafego_baixo(T1),trafego_
medio(T2)), trafego_alto(T3)),
trafego_baixo(T4)), tempo_longo);
uc5 =
min(min(min(min(trafego_medio(T1),trafego_
alto(T2)), trafego_alto(T3)),
trafego_alto(T4)), tempo_longo);
uc6 =
min(min(min(min(trafego_medio(T1),trafego_
baixo(T2)), trafego_baixo(T3)),
trafego_medio(T4)), tempo_medio);
uc7 =
min(min(min(min(trafego_medio(T1),trafego_
```

```
medio(T2)), trafego_muito_alto(T3)),
trafego_baixo(T4)), tempo_muito_longo);
uc8 =
min(min(min(min(trafego_medio(T1),trafego_
alto(T2)), trafego_baixo(T3)),
trafego_muito_alto(T4)), tempo_longo);
uc9 =
min(min(min(min(trafego_alto(T1),trafego_m
edio(T2)), trafego_alto(T3)),
trafego_medio(T4)), tempo_longo);
uc10 =
min(min(min(min(trafego_alto(T1),trafego_a
lto(T2)), trafego_medio(T3)),
trafego_medio(T4)), tempo_longo);
uc11 =
min(min(min(min(trafego_alto(T1),trafego_b
aixo(T2)), trafego_baixo(T3)),
trafego_alto(T4)), tempo_longo);
uc12 =
min(min(min(min(trafego_alto(T1),trafego_m
edio(T2)), trafego_baixo(T3)),
trafego_baixo(T4)), tempo_longo);
uc13 =
min(min(min(min(trafego_muito_alto(T1),tra
fego_muito_alto(T2)), trafego_baixo(T3)),
trafego_muito_alto(T4)),
tempo_muito_longo);
uc14 =
min(min(min(min(trafego_muito_alto(T1),tra
fego_baixo(T2)), trafego_medio(T3)),
trafego_baixo(T4)), tempo_muito_longo);
uc15 =
min(min(min(min(trafego_muito_alto(T1),tra
fego_medio(T2)), trafego_muito_alto(T3)),
trafego_alto(T4)), tempo_muito_longo);
uc16 =
min(min(min(min(trafego_muito_alto(T1),tra
fego_baixo(T2)), trafego_alto(T3)),
trafego_baixo(T4)), tempo_muito_longo);
uc17 =
min(min(min(min(trafego_medio(T1),trafego_
muito_alto(T2)), trafego_baixo(T3)),
trafego_alto(T4)), tempo_medio);
uc18 =
min(min(min(min(trafego_baixo(T1),trafego_
medio(T2)), trafego_muito_alto(T3)),
trafego_medio(T4)), tempo_muito_longo);
uc19 =
min(min(min(min(trafego_baixo(T1),trafego_
alto(T2)), trafego_baixo(T3)),
trafego_baixo(T4)), tempo_curto);

uc20 =
min(min(min(min(trafego_baixo(T1),trafego_
baixo(T2)), trafego_baixo(T3)),
trafego_baixo(T4)), tempo_curto);
uc21 =
min(min(min(min(trafego_baixo(T1),trafego_
baixo(T2)), trafego_medio(T3)),
trafego_alto(T4)), tempo_longo);
uc22 =
min(min(min(min(trafego_baixo(T1),trafego_
medio(T2)), trafego_medio(T3)),
trafego_alto(T4)), tempo_longo);
uc23 =
min(min(min(min(trafego_baixo(T1),trafego_
medio(T2)), trafego_alto(T3)),
trafego_baixo(T4)), tempo_medio);
uc24 =
min(min(min(min(trafego_medio(T1),trafego_
alto(T2)), trafego_alto(T3)),
trafego_alto(T4)), tempo_longo);
uc25 =
min(min(min(min(trafego_medio(T1),trafego_
baixo(T2)), trafego_baixo(T3)),
trafego_medio(T4)), tempo_medio);
uc26 =
min(min(min(min(trafego_medio(T1),trafego_
medio(T2)), trafego_muito_alto(T3)),
trafego_baixo(T4)), tempo_medio);
uc27 =
min(min(min(min(trafego_medio(T1),trafego_
alto(T2)), trafego_baixo(T3)),
trafego_muito_alto(T4)),
tempo_muito_longo);
uc28 =
min(min(min(min(trafego_alto(T1),trafego_m
edio(T2)), trafego_alto(T3)),
trafego_medio(T4)), tempo_medio);
uc29 =
min(min(min(min(trafego_alto(T1),trafego_a
lto(T2)), trafego_medio(T3)),
trafego_medio(T4)), tempo_longo);
uc30 =
min(min(min(min(trafego_alto(T1),trafego_b
aixo(T2)), trafego_baixo(T3)),
trafego_alto(T4)), tempo_longo);
uc31 =
min(min(min(min(trafego_alto(T1),trafego_m
edio(T2)), trafego_baixo(T3)),
trafego_baixo(T4)), tempo_medio);
uc32 =
min(min(min(min(trafego_muito_alto(T1),tra
fego_muito_alto(T2)), trafego_baixo(T3)),
trafego_muito_alto(T4)),
tempo_muito_longo);
uc33 =
min(min(min(min(trafego_muito_alto(T1),tra
fego_baixo(T2)), trafego_medio(T3)),
trafego_baixo(T4)), tempo_curto);
uc34 =
min(min(min(min(trafego_muito_alto(T1),tra
fego_medio(T2)), trafego_muito_alto(T3)),
trafego_alto(T4)), tempo_longo);
uc35 =
min(min(min(min(trafego_muito_alto(T1),tra
fego_baixo(T2)), trafego_alto(T3)),
trafego_baixo(T4)), tempo_curto);
uc36 =
min(min(min(min(trafego_medio(T1),trafego_
muito_alto(T2)), trafego_baixo(T3)),
trafego_alto(T4)), tempo_muito_longo);
uc37 =
min(min(min(min(trafego_baixo(T1),trafego_
medio(T2)), trafego_muito_alto(T3)),
trafego_medio(T4)), tempo_medio);
uc38 =
min(min(min(min(trafego_baixo(T1),trafego_
```

```
alto(T2)), trafego_baixo(T3)),
trafego_baixo(T4)), tempo_longo);


% Defuzzificação

Y1=max([uc1 ;uc2 ;uc3 ;uc4 ;uc5; uc6 ;uc7;
uc8 ;uc9 ;uc10 ;uc11 ;uc12 ;uc13 ;uc14
;uc15 ;uc16 ;uc17 ;uc18 ;uc19]);

%figure
plot(tempo_curto); hold on;
plot(tempo_medio); plot(tempo_longo);
plot(tempo_muito_longo); area(Y1,
'FaceColor', palette);
axis([0 100 0 1]);
legend('Curto', 'Medio', 'Longo','Muito
Longo', 'Resultado S1/S3','Location',
'Southeast');
hold off;

Y2=max([uc20 ;uc21 ;uc22 ;uc23 ;uc24; uc25
;uc26; uc27 ;uc28 ;uc29 ;uc30 ;uc31 ;uc32
;uc33 ;uc34 ;uc35 ;uc36 ;uc37 ;uc38]);

%figure
plot(tempo_curto); hold on;
plot(tempo_medio); plot(tempo_longo);
plot(tempo_muito_longo); area(Y2,
'FaceColor', palette);
axis([0 100 0 1]);
legend('Curto', 'Medio', 'Longo','Muito
Longo','Resultado S2/S4', 'Location',
'Southeast');
hold off;

% Saída
% Calculo utilizando o método do centro de
area
 w1 = Y1.*x/Y1

 w2 = Y2.*x/Y2

end


%% Funcoes de pertinencia implementadas


function tri = TriangularMF(x, a, b, c)

    for i = 1:numel(x)
        if x(i) < a
            tri(i) = 0;
        elseif a <= x(i) && x(i) <= b
            tri(i) = (x(i) - a) / (b - a);
        elseif b < x(i) && x(i) <= c
            tri(i) = 1 - (x(i) - b) / (c -
b);
        elseif x(i) > c
            tri(i) = 0;
        end
    end
end


function gauss = GaussMF(x, sigma, m)
    for i=1:numel(x)
        gauss(i)=exp(-1/2*(1/sigma*(x(i)-
m))^2);
    end
end


function trap = TrapezoidalMF(x,a,b,c,d)
    for  i= 1:numel(x)
        if x(i) < a
            trap(i) = 0;
        elseif a <= x(i) && x(i) <= b
            trap(i) = (x(i) - a) / (b -
a);
        elseif b < x(i) && x(i) <= c
            trap(i) = 1;
        elseif c < x(i) && x(i) <= d
            trap(i) = 1 - (x(i) - c) / (d-
c);
        elseif d < x(i)
            trap(i)=0;
        end
    end
end
```