UNIVERSIDADE FEDERAL DE MINAS GERAIS

Tópicos em Inteligência Artificial - Sistemas Nebulosos

Exercício Computacional 2 - Classificador nebuloso de Flores

Luiza Agostinho Pena - 2014034855
Luiz Henrique Silva Lelis - 2014034847

02/05/2017

Sumário

Lista de Figuras

1 Introdução

Na teoria clássica um dado elemento do universo (domínio) pertence ou não pertence ao referido conjunto. Na teoria dos conjuntos "fuzzy" (ou nebulosos) existe um grau de pertinência de cada elemento a um determinado conjunto. Esse grau de pertinência é determinado através de uma função denominada Função de Pertinência. A lógica fuzzy estende a lógica booleana, ao invés de permitir só dois valores possíveis (verdadeiro ou falso), ela permite uma gama infinita de valores.

O presente relatório propõe implementar um classificador Fuzzy de flores que classifica os elementos a partir de dois parâmetros. O primeiro parâmetro é o comprimento da pétala, e o outro é a sua largura . Os elementos podem tomar três classes distintas: Iris Setosa, Iris Versicolor ou Iris Virginica. Essas classes correspondem a três tipos diferentes de flores.

Um sistema nebuloso pode ser dividido em cinco etapas de execução: Fuzzificação, Aplicação dos operadores Fuzzy, Operador de Implicação, Combinação de todas as saídas Fuzzy possíveis e por fim, a etapa de Defuzzificação. O presente relatório divide a documentação em duas grandes etapas: Treinamento do Classificador e Teste do Classificador. A primeira etapa é subdividida nas cinco etapas de execução de um sistema nebuloso. Já a segunda grande etapa é dividida em: Teste das Classes Treinadas e Teste final do Classificador (Validando a Implementação).

A etapa de Fuzzyficação consiste na atribuição de uma função de pertinência para cada valor de entrada, que permite obter o grau de verdade da proposição. A segunda etapa (Aplicação dos operadores Fuzzy) é responsável por definir o grau máximo e mínimo de pertinência do conjunto a partir de operadores. O terceiro passo é aplicar o operador de implicação, usado para definir o peso no resultado e remodelar a função, ou seja, o terceiro consiste em criar a hipótese de implicação. No quarto passo ocorre a combinação de todas as saídas em um único conjunto fuzzy. E por fim, a defuzzyficação consiste em retornar os valores, obtendo um valor numérico dentro da faixa estipulada pela lógica fuzzy.

2 Objetivos

Implementar o classificador nebuloso descrito em (Ishibuchi, 2001)[1]. Deve-se validar a implementação na base de dados Iris.

3 Resultados

3.1 Treinamento do Classificador

3.1.1 Fuzzificação

O sistema foi subdivido em funções de pertinência triangulares no eixo X e eixo Y de um Grid Fuzzy. Essas funções são igualmente espaçadas e se encontram na mesma quantidade no eixo X e Y, a quantidade de funções de pertinência em cada eixo é definida por um hiperparâmetro λ que também será chamado de nμ ao longo do relatório (onde μ corresponde à FP e n à quantidade de FP).

Dado um elemento de entrada E, onde E é uma flor com dois parâmetros que a classifica, E pode ser interpretado como um vetor de duas posições. O valor da primeira posição é o comprimento da pétala e o valor na segunda posição é a largura da pétala. Dessa forma temos:

$$\begin{aligned} \mu_{comprimento}(E) &= trimf(E[1]) \\ \mu_{largura}(E) &= trimf(E[2]) \end{aligned} \tag{1}$$

A função de pertinência depende de um fator Δ que definirá o espaçamento dos parâmetros das funções triangulares. O valor escolhido para esse parâmetro Δ segue abaixo, junto à forma com que ele foi atribuído às funções de pertinência triangulares:

$$\begin{aligned} \Delta_{comprimento} &= \frac{Max_{comprimento} - Min_{comprimento}}{\lambda - 1} \\ \Delta_{largura} &= \frac{Max_{largura} - Min_{largura}}{\lambda - 1} \\ FP_X &= [-\Delta_{comprimento}, 0, +\Delta_{comprimento}] \\ FP_Y &= [-\Delta_{largura}, 0, +\Delta_{largura}] \end{aligned} \tag{2}$$

Segue abaixo a parte do código correspondente à etapa de Fuzzyficação:

```
% CONJUNTO DE TREINEMNTO: x_train e y_train
x_train = reshape(textread('x_train.txt', '%f'),2,105);
x_train = x_train';
size_in = size(x_train);

x_lenght = x_train(:, 1);
x_width = x_train(:, 2);

% Verifica a faixa de valores das entradas e adiciona uma
    margem de erro
min_lenght = 0;
max_lenght = int64(max(x_lenght) + 0.5*max(x_lenght));
min_width = 0;
max_width = int64(max(x_width) + 0.2*max(x_width));

y_train = reshape(textread('y_train.txt', '%f'),1,105);
y_train = y_train';

fig = 3;
```

```
% ********************************************************
% ************** TREINAMENTO DO CLASSIFICADOR *************
% ********************************************************
% Define os Hiperparametros de entrada do classificador
MiMin = 1;
MiMax = 10;
for nMi=1:MiMax

% Calcula espacamento uniforme para as funcoes de
    pertinencia

lenght_spacing = (max_lenght - min_lenght)/(nMi-1);
width_spacing = (max_width - min_width)/(nMi-1);

% As funcoes de pertinencia serao triagulares.
% A primeira funcao de pertinencia comecara com centro em
    zero e
% as consequentes terao seu centro espacado conforme
    definido no
% spacing calculado.

% Sao calculadas as pertinencias para todas as entradas e,
    para cada
% entrada, eh reservada uma linha na matriz desse resultado,
     sendo cada
% coluna dessa linha o resultado daquela entrada para a
    respectiva
% funcao de pertinencia
for x=1:size_in(1)
    for m=1:nMi
        lenght_a(m) = double((m-2)*lenght_spacing);
        lenght_b(m) = double((m-1)*lenght_spacing);
        lenght_c(m) = double((m)*lenght_spacing);
        mis_lenght(x, m) = trimf(x_lenght(x), [lenght_a(m)
            lenght_b(m) lenght_c(m)]);

        width_a(m) = double((m-2)*width_spacing);
        width_b(m) = double((m-1)*width_spacing);
        width_c(m) = double((m)*width_spacing);
        mis_width(x, m) = trimf(x_width(x), [width_a(m)
            width_b(m) width_c(m)]);
    end
end
```

3.1.2 Aplicação dos operadores Fuzzy

A operação aplicada aos elementos Fuzzy é a operação de interseção entre as funções de pertinências referente àquela área do Grid de cada eixo. Essa operação foi escolhida por também ter sido usada em (Ishibuchi, 2001)[1].

Assim sendo, a pertinência final dessa entrada E será dada por:

$$\mu_E = \mu_{comprimento}(E) \cdot \mu_{largura}(E) \tag{3}$$

Segue abaixo a parte do código correspondente à etapa de aplicação dos operadores Fuzzy:

```
% A Pertinencia do vetor de entrada sera o produto das
    funcoes de
% pertinencia de largura e altura.
```

```
mx = 1;
my = 1;
k = 1;
for i=1:size_in(1)
    k = 1;
    for my = 1:nMi
        for mx = 1:nMi
            mis(i, k) = mis_lenght(i,mx)*mis_width(i,my);
            k = k+1;
        end
    end
end
```

3.1.3 Operador de Implicação

Essa etapa corresponde as regras IF-THEN citadas em (Ishibuchi, 2001)[1]. A entrada E será classificada em uma determinada região R_j . Como essa entrada provavelmente terá valores de pertinência diferentes de zero em mais de uma região, um grau de certeza C_j será utilizado para determinar qual região e seu consequente será atribuído àquela entrada.

O grau de certeza será calculado no procedimento de treinamento do classificador, e aplicado posteriormente junto às funções de pertinência para a classificação das entradas. Em relação à regra geral, o consequente de uma região R_j será dado pelas amostras que tiverem pertinência diferente de zero naquela região. Dada uma região R_j, serão contabilizadas quantas amostras de cada classe tem pertinência diferente de zero naquela região, assim como a soma dessas pertinências de cada classe de amostras, dada por:

$$\beta_k = \sum_{E \ \epsilon \ Classe_k} \mu_i(E) \tag{4}$$

Onde β_k será usado para o cálculo do grau de certeza da classificação daquela região.

Segue abaixo a parte do código correspondente à etapa de aplicação dos operadores de Implicação:

```
% Classifica as nMi regioes entre flor 1, 2 ou 3
beta = zeros (3,nMi^2);
for j=1:(nMi^2)
    n1 = 0;
    n2 = 0;
    n3 = 0;
    for i=1:size_in(1)
        if (mis(i,j) ~= 0)
            if y_train(i) == 1
                n1 = n1+1;
                beta(1,j) = beta(1,j) + mis(i,j);
            elseif y_train(i) == 2
                n2 = n2+1;
                beta(2,j) = beta(2,j) + mis(i,j);
            else
                n3 = n3+1;
                beta(3,j) = beta(3,j) + mis(i,j);
            end
        end
    end
```

3.1.4 Combinação das saídas Fuzzy + Defuzzificação

A definição do grau de certeza da regra se dá a partir de β_k. Uma regra R_i classificará uma entrada como C_i com um grau de certeza CF_i, e que temos 3 classes possíveis de flores e que a classificação daquela área é, por exemplo, 1 -¿ Iris Setosa. Esse grau de certeza será calculado como:

$$CF_i = \frac{\beta_1(R_i) - \beta_2(R_i) - \beta_3(R_i)}{\beta_1(R_i) + \beta_2(R_i) + \beta_3(R_i)} \tag{5}$$

Para as demais classes, o seu β_k fica positivo na parte de cima da fração e os demais negativos.

A Regra do Resultante se dá a partir de uma entrada E. Dada a entrada E, define-se qual é seu consequente pela área que deter o maior valor de μ para aquela entrada, pesando-se seu grau de certeza. Isso será melhor explicado em DEFINIÇÃO DA REGRA RESULTANTE APLICADA A UMA ENTRADA.

Definição da Regra Resultante Aplicada a uma Entrada

A regra R_j resultante, aplicada a uma determinada entrada E, será definida por:

$$R_j : \mu_j(E) = max(\mu_j(E) \cdot C_j : j = 1, 2, 3, ..., N) \tag{6}$$

Onde:

- N é $n\mu^2$, ou seja, dado o número de subdivisões em cada eixo $n\mu^2$, é a quantidade de áreas subdivididas no Grid Fuzzy que define a quantidade de regras;
- $\mu_j(E)$ é o valor de pertinência resultante daquela entrada para cada área subdividida do Grid Fuzzy. Ou seja, para cada uma das regras R_j (conforme explicado em Fuzzificação e Aplicação dos operadores Fuzzy) só que aplicado ao algoritmo resultante do treinamento;
- C_j é o grau de certeza para aquela regra R_j, valores definidos previamente no treinamento.

Definição da Saída e Erro de Classificação

Definida qual é a regra consequente para uma determinada entrada E, a saída do problema será o consequente r_j da regra R_j, com uma certeza C_j, definidos no treinamento.

O erro da classificação será definido por:

$$e = \frac{n_{erros}}{n_{amostras}} \tag{7}$$

Ou seja, será verificado o resultado esperado do conjunto de teste disponibilizado e, caso esse valor seja diferente do resultado do classificador para uma entrada, será contabilizado um erro. O valor do erro final será a porcentagem de classificações erradas sobre o número total de flores classificadas.

3.2 Teste do Classificador

3.2.1 Teste das Classes Treinadas

Para esse teste inicial, foi plotado o conjunto de treinamento no inicio do programa. Feito isso, essa mesma matriz foi colocada de entrada no classificador já treinado para observar o seu desempenho. O resultado do conjunto de treino (Resultado esperado), o resultado obtido para 3 funções de pertinência ($n\mu = 3$) e o resultado obtido para 10 funções de pertinência ($n\mu = 10$) seguem abaixo para análise:

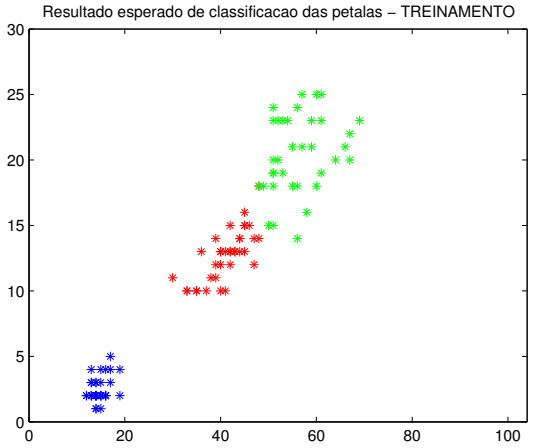


Figura 1: Resultado do Conjunto de Treino (Esperado)

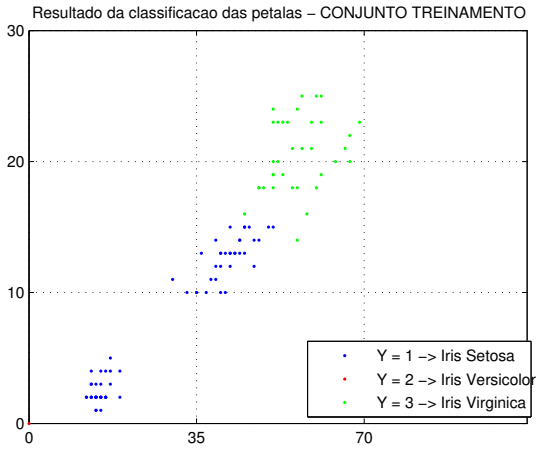


Figura 2: Resultado da Matriz Treino com $n\mu = 3$

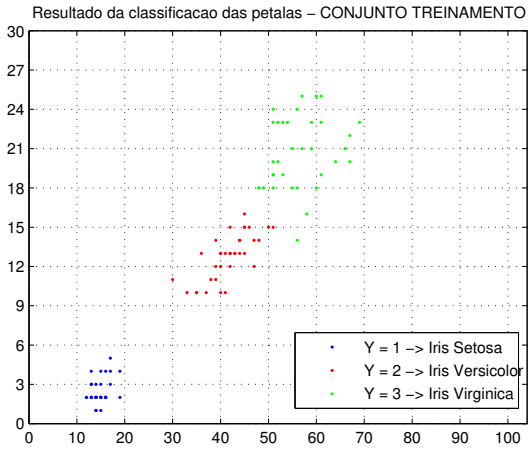


Figura 3: Resultado da Matriz Treino com $n\mu = 10$

3.2.2 Teste final do Classificador (Validando a Implementação)

Para esse teste final, foi plotado o conjunto de teste no inicio do programa. Feito isso, essa mesma matriz foi colocada de entrada no classificador já treinado para observar o seu desempenho. O resultado do conjunto de teste (Resultado esperado), o resultado obtido para 3 funções de pertinência ($n\mu = 3$) e o resultado obtido para 10 funções de pertinência ($n\mu = 10$) seguem abaixo para análise:

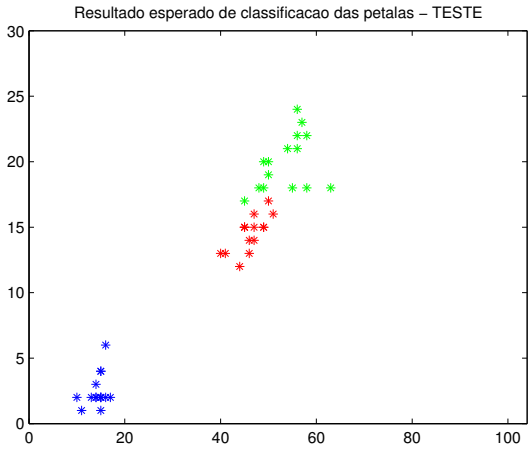


Figura 4: Resultado do Conjunto de Teste (Esperado)

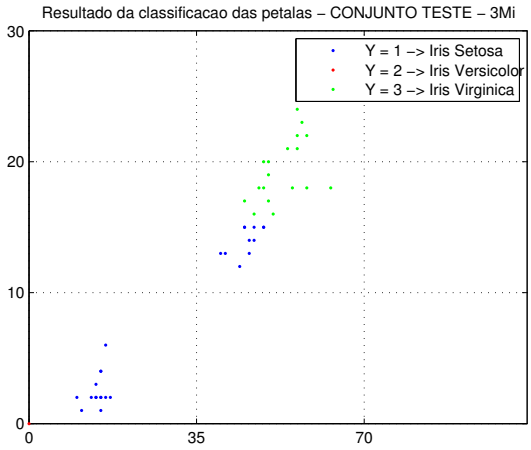


Figura 5: Resultado da Matriz Teste com $n\mu = 3$

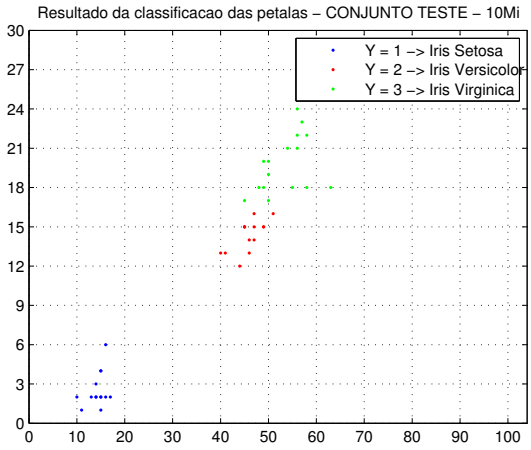
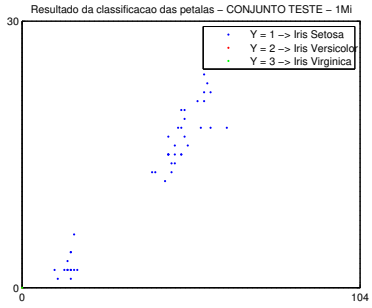
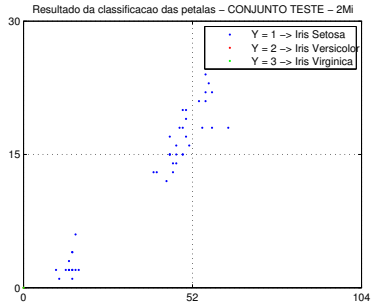


Figura 6: Resultado da Matriz Teste com $n\mu = 10$

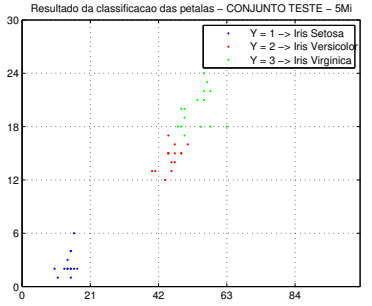
Para analisar o comportamento do classificador como um todo, a quantidade de funções de pertinência foram sendo alteradas (quantidade de $n\mu$). Foi possível analizar o comportamento tanto para o treino quanto para o teste. Segue abaixo os resultados obtidos para a classificação da matriz teste para $n\mu$ variando de 1 a 10:



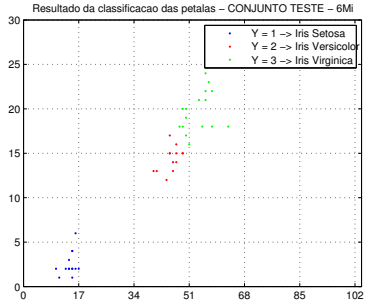
(a) Classificação Matriz Teste $n\mu = 1$



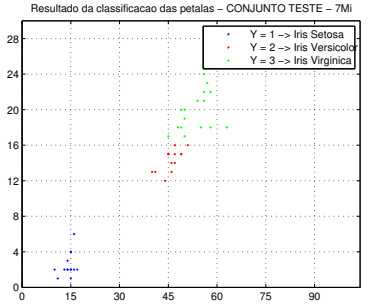
(b) Classificação Matriz Teste $n\mu = 2$



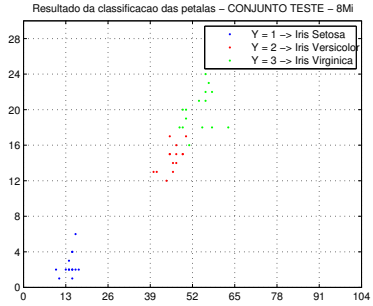
(c) Classificação Matriz Teste $n\mu = 5$



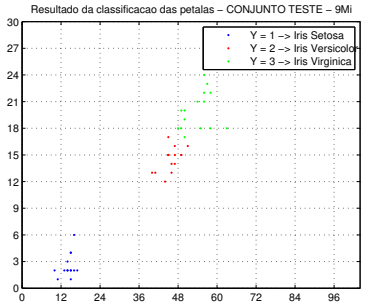
(d) Classificação Matriz Teste $n\mu = 6$



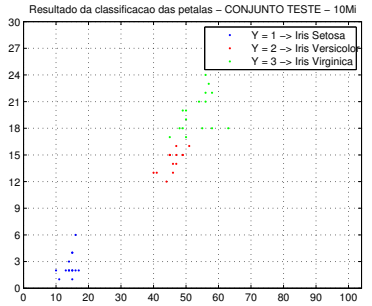
(e) Classificação Matriz Teste $n\mu = 7$



(f) Classificação Matriz Teste $n\mu = 8$



(g) Classificação Matriz Teste $n\mu = 9$



(h) Classificação Matriz Teste $n\mu = 10$

Por fim, foi plotado o gráfico do erro em função do número de funções de pertinência variando de 1 a 50 FPs. O resultado segue abaixo:

Figura 8: Gráfico do erro em função do número de FPs ($0 < n\mu < 51$)

Considerando uma configuração mais realista para o sitema, foi criado um gráfico do erro em função do número de funções de pertinência variando de 1 a 10 FPs. O resultado segue abaixo:

Figura 9: Gráfico do erro em função do número de FPs ($0 < n\mu < 11$)

4 Discussão

Foi possível observar uma aproximação muito grande do resultado esperado ao colocarmos 10 funções de pertinência no sistema ($n\mu = 10$) tanto para o conjunto de Treino, quanto para o conjunto de Teste. Outra característica importante do sistema é que por ter 3 classes, quando $n\mu$ possui um valor menor que 3, a classificação perde seu sentido. Por isso, observa-se um resultado bem longe do esperado com $n\mu = 1$ e $n\mu = 2$.

Outros pontos com classificações bem próximas do esperado foram obtidos em $n\mu = 5$ e $n\mu = 7$. Isso tudo pode ser confirmado através da análise dos gráficos do erro em função do número de FPs. Foi possível observar pequenos valores de erro para esses pontos citados (5,7,10). Ao aumentar demais o número de FPs, a classificação também perde o seu sentido, o erro tende a crescer, e as classes não serão atribuídas de forma correta.

O número de funções de pertinência escolhido para a classificação influencia diretamente no grid Fuzzy que é criado. Por esse motivo, ao aumentarmos demais o valor de $n\mu$, para o atual conjunto de teste (com 105 elementos), a classificação tende a piorar porque a área correspondente a certa classe ficará muito subdividida. Assim sendo, o erro cresce ao aumentar demais o valor de $n\mu$. Por outro lado, para uma quantidade muito pequena de funções de pertinência, as áreas correspondentes às classes abrangem uma área muito grande no grid fuzzy, o que prejudica a diferenciação das classes pelo algoritmo.

Analisando o exposto acima, observa-se resultados bastante satisfatórios. Sendo assim, os objetivos propostos foram resolvidos, possibilitando que o exercício computacional fosse concluído com êxito.

Referências

[1] Hisao Ishibuchi and Tomaharu Nakashima. ***Effect of Rule Weights in Fuzzy Rule-Based Classification Systems***. IEEE TRANSACTIONS ON FUZZY SYSTEMS, VOL. 9, NO. 4, AUGUST 2001.