UNIVERSIDADE FEDERAL DE MINAS GERAIS

Tópicos em Inteligência Artificial - Sistemas Nebulosos

Exercício Computacional 4 - Estimar parâmetros com GDE

Luiz Henrique Silva Lelis - 2014034847

04/05/2017

Sumário

1 Introdução **2**

2 Objetivos **2**

3 Metodologia **3**

4 Resultados **6**

5 Discussão **9**

Lista de Figuras

1 Comparativo sinc(x) Real Estimado - Inicial e Final 6
1a Primeira iteração 6
1b Última iteração 6
2 Comparativo sinc(x) Real Estimado - Completo 7
2a Iteração 2 7
2b Iteração 5 7
2c Iteração 10 7
2d Iteração 15 7
2e Iteração 30 7
2f Iteração 40 7
2g Iteração 60 7
2h Iteração 85 7
3 Gráfico das funções de pertinência iniciais e finais 8
4 Gráfico das Retas y_k 8
5 Gráfico do custo J em função da época de treinamento 8

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.

De acordo com (Mamdani, 1973)[1], um sistema nebuloso pode ser dividido em cinco etapas de execução: Fuzzificação, Aplicação dos operadores Fuzzy, Aplicação da Inferência, Combinação de todas as saídas Fuzzy possíveis e, por fim, a etapa de Defuzzificação. 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 inferência, 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.

Regras do tipo Takagi-Sugeno são usadas na etapa de inferência citada anteriormente. Esse documento descreve como o algoritmo do gradiente descendente estocástico pode ser utilizado para estimar os parâmetros de um conjunto de regras do tipo Takagi-Sugeno.

2 Objetivos

Estimar os parâmetros de um conjunto de regras do tipo Takagi-Sugeno a partir do uso do gradiente descendente estocástico.

3 Metodologia

A primeira ação realizada pelo algoritmo é criar as entradas e a função de saída real (esperada). As dimensões dessa entrada correspondem às variáveis n e d (numero de pontos de entrada e número de dimensões respectivamente). Essas variáveis são armazenadas pois serão responsáveis por definir as dimensões de várias outras matrizes.

Feito isso, os parâmetros das regras Takagi-Sugeno são inicializados respeitando as definições. Os valores dos centros das funções de pertinência foram definidos como valores aleatórios com distribuição uniforme no intervalo em que a variável correspondente possui amostras. Ou seja, $c_{lk} = U[min(x_l), max(x_l)]$. Os valores das dispersões das funções de pertinência foram definidos como valores aleatórios positivos com distribuicão uniforme no intervalo unitário, ou seja, $\sigma_{lk} = U[0, 1]$. Por fim, os valores dos parâmetros do consequente foram definidos como valores aleatórios no intervalo $[-1, 1]$. O código correspondente a essa parte do algoritmo segue abaixo:

```
% Funcao sinc real (Usa 100 pontos de 0 a 2pi)
x = linspace(0,2*pi,100)';
[n,d] = size(x);
yReal = sinc(x);

% Pega os limites da matriz x
minX = min(x);
maxX = max(x);

% Estimativa inicial para os parametros
sig = rand(d,m);
c = (maxX-minX).*rand(d,m)+minX;
p = 2.*rand(d,m)-1;
q = 2.*rand(1,m)-1;
```

Após a inicialização das matrizes, o principal loop do algoritmo é iniciado. Esse loop só é encerrado quando a função de custo é minimizada o suficiente para atingir sua condição de parada ($< 1^{-3}$). A funcao a ser minimizada segue abaixo:

$$J(c_{lk}, \sigma_{lk}, p_{lk}, q_k) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{1}$$

Todo o processo do algoritmo do presente relatório é feito por amostra. Isso significa que as ações a serem tomadas são feitas de ponto a ponto da entrada x. Para isso, foi criado um novo loop para iterar sobre as entradas. Cada elemento da entrada é percorrido para calcular: a função de pertinência, a função de consequentes, a saída do modelo (estimado), as derivadas parciais para estimar os parâmetros desse modelo, e por fim, a atualização dos parâmetros do gradiente descendente estocástico (GDE). Todas essas etapas são detalhadas abaixo.

No modelo considerado, as funções de pertinência escolhidas para aos conjuntos nebulosos dos antecedentes de cada regra foram Gaussianas e possuem a seguinte forma:

$$\mu_{A_{lk}}(X_{il}) = exp\left(-\frac{1}{2}\frac{(x_{il} - c_{lk})^2}{\sigma_{lk}^2}\right) \tag{2}$$

Foi utilizada a função *gaussmf()* nativa do MATLAB para criar as FPs. Segue abaixo a parte do código correspondente ao cálculo função de pertinência:

```
function mi = pertinenceFunc(x, sig, c, i, m, d)

    % Calcula a funcao de pertinencia
    mi = zeros(d,m);
    for k=1:m
        mi(k) = gaussmf(x(i), [sig(k) c(k)]);
    end

end
```

A função dos consequentes é dada por $y_k = p_{1k}x_{i1} + p_{2k}x_{i2} + ... + p_{dk}x_{id} + q_k$. Como no modelo atual o número de entradas é igual ao número de regras, temos uma função consequente para cada antecedente. Dessa forma, a equação dos consequentes é dada por $y_k = p_{1k}x_{i1} + q_k$.

Além disso, como as entradas do modelo atual possuem uma única dimensão, o ω será igual a μ. Essa característica influencia no cálculo da saída do modelo (y estimado). A equação do y estimado agora é dado por:

$$\hat{y}_i = \frac{\sum_{k=1}^{m} \mu_k y_k}{\sum_{k=1}^{m} \mu_k} \tag{3}$$

Com essas alterações o código ficou conforme explicitado abaixo:

```
% Calcula o consequente da amostra atual
y = p.*x(i) + q;

% Calcula a saida do modelo (estimado)
yEst = sum(mi.*y)/(sum(mi));
```

Para estimar os parâmetros do atual modelo, usou-se o algoritmo do gradiente descendente estocástico. Para isso, foi necessário encontrar a derivada parcial da função de custo em relação a cada um dos parâmetros do modelo. Usando o que já foi dito anteriormente, as derivadas parciais ficaram da seguinte forma:

$$\frac{\partial J}{\partial c_{lk}} = \left(-2\sum_{i=1}^{n}(y_i - \hat{y}_i)\right)\left(\frac{y_k - \hat{y}_i}{\sum_{k=1}^{m}\mu_k}\right)\left(\mu_k\frac{(x_l - c_{lk})}{\sigma_{lk}^2}\right) \tag{4}$$

$$\frac{\partial J}{\partial \sigma_{lk}} = \left(-2\sum_{i=1}^{n}(y_i - \hat{y}_i)\right)\left(\frac{y_k - \hat{y}_i}{\sum_{k=1}^{m}\mu_k}\right)\left(\mu_k\frac{(x_l - c_{lk})^2}{\sigma_{lk}^3}\right) \tag{5}$$

$$\frac{\partial J}{\partial p_{lk}} = \left(-2\sum_{i=1}^{n}(y_i - \hat{y}_i)\right)\left(\frac{\mu_k}{\sum_{k=1}^{m}\mu_k}\right)(x_l) \tag{6}$$

$$\frac{\partial J}{\partial q_{lk}} = \left(-2\sum_{i=1}^{n}(y_i - \hat{y}_i)\right)\left(\frac{\mu_k}{\sum_{k=1}^{m}\mu_k}\right) \tag{7}$$

Sabe-se que o algoritmo do GDE utiliza uma estimativa do vetor gradiente, baseando-se em apenas uma amostra. Dessa forma, nesse algoritmo, a expressão da derivada parcial da função de custo em relação a $\hat{y}_i$ foi definida como:

$$\frac{\partial J}{\partial \hat{y}_i} = -2(y_i - \hat{y}_i) \tag{8}$$

Feito isso, o algoritmo segue com a atuzalização dos parâmetros do modelo por amostra. A equação da atualização é a mesma para todos os parâmetros, por isso, foi criada uma função que é chamada por cada um dos parâmetros. Segue abaixo como foi desenvolido no código:

```
function newParam = updateParameter(param, alpha,
    partialDiff)

    % Estima os parametros do modelo

    newParam = param - alpha*partialDiff;

end
```

Por fim, fora do loop das amostras de entrada, calcula-se novamente as funções que são utilizadas pela função de custo. A função de pertinência, a consequente e a saída estimdada são recalculadas nesse ponto com os parâmetros atuzalizados. Por fim, calcula-se a função de custo e os gráficos são plotados.

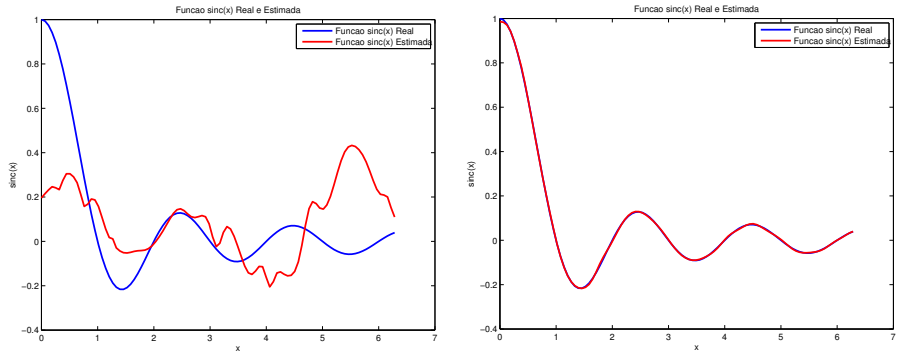
4 Resultados

Para o modelo atual, foram utilizadas 100 amostras de entrada no intervalo de 0 a 2π. Para o custo inicial J foi atribuído 20. Para o número de regras foi atribuído 100, dessa forma para cada antecedente existia um consequente (um x e um y por regra). Nessa configuração, o número de épocas (iterações) foi de 87. Segue abaixo uma tabela contendo os valores dessas variáveis:

Variável	**Valor atribuído**
n (Amostras de entrada)	100
J (Custo Inicial)	20
m (Número de Regras)	100
Número de épocas	87

Tabela 1: Variáveis importantes e seus valores atribuídos

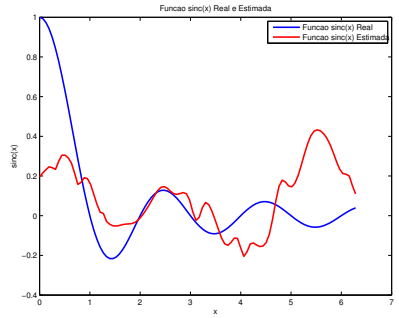
Na configuração atual, o algoritmo imprime o gráfico da saída real e da saída estimada à medida que são feitas novas iterações. A primeira iteração é salva no formato ".eps" na pasta "resultados" e as próximas são salvas de cinco em cinco. Pouco depois é impresso o gráfico das retas yk. Depois da função convergir, são impressos os gráficos do custo J em função da época de treinamento e das funções de pertinência iniciais (aleatórios) e após a convergência. Segue abaixo um comparativo da saída estimada (yhat) inicial e a saída real na primeira e última iteração (respectivamente):



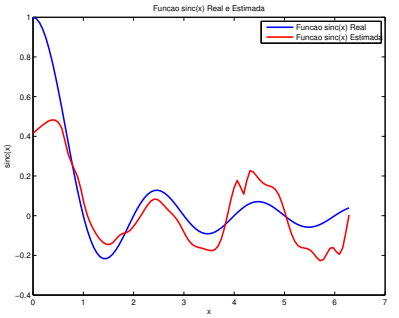
(a) Primeira iteração (b) Última iteração

Figura 1: Comparativo sinc(x) Real Estimado - Inicial e Final

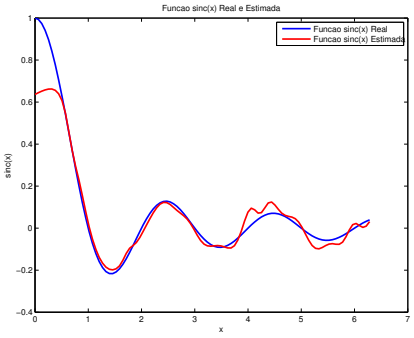
Segue abaixo um comparativo mais completo, agora contemplando várias iterações diferentes. À medida que o número de iterações aumenta, a função chega mais perto de convergir:



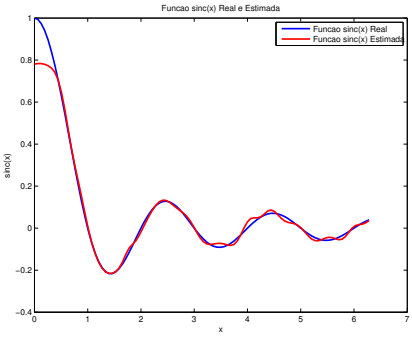
(a) Iteração 2



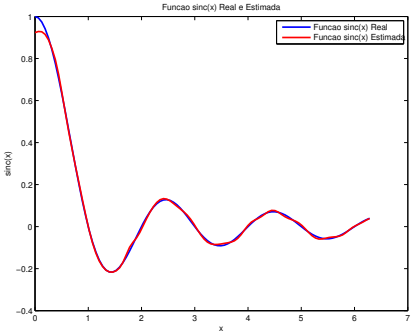
(b) Iteração 5



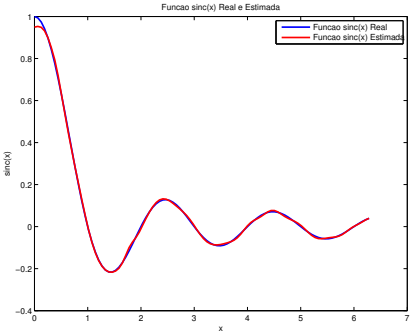
(c) Iteração 10



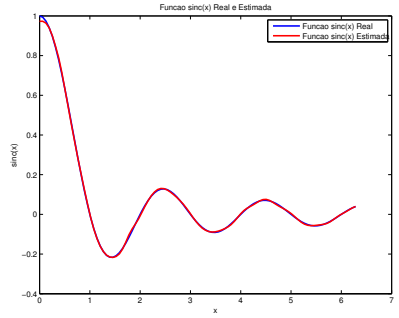
(d) Iteração 15



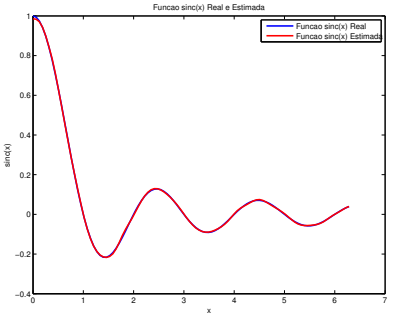
(e) Iteração 30



(f) Iteração 40



(g) Iteração 60



(h) Iteração 85

Figura 2: Comparativo sinc(x) Real Estimado - Completo

Segue abaixo os gráficos das funções de pertinência iniciais e finais, das retas y_k e por fim da função de custo versus épocas de treinamento:

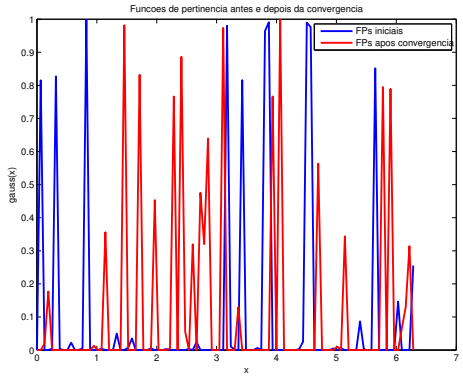


Figura 3: Gráfico das funções de pertinência iniciais e finais

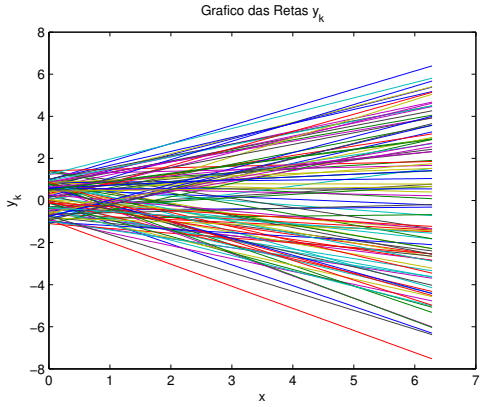


Figura 4: Gráfico das Retas y_k

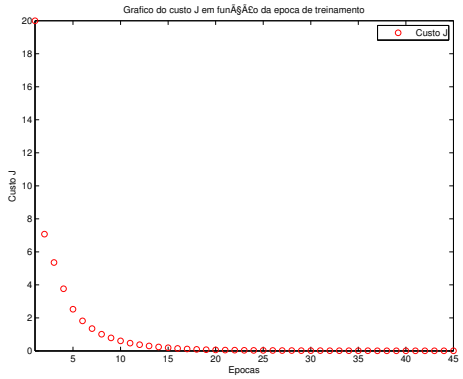


Figura 5: Gráfico do custo J em função da época de treinamento

5 Discussão

Durante os experimentos observou-se uma queda no resultado da função objetivo, o que era previso, já que a função deveria ser minimizada. Ou seja, o saída do modelo (yHat) convergiu de forma coerente com a função sinc(x).

Além disso o valor encontrado para o RMSE (= 0,0032) é bastante satisfatório pois indica quão próximo o estimado está do real (a diferença é muito próxima a zero). Comparando os resultados obtidos com a função yReal(sinc(x)), observa-se uma semelhança muito grande entre os gráficos. Isso é um indicativo de que o algoritmo foi implementado de forma correta.

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] Mamdani, E.H. ***Applications of fuzzy algorithm for control of simple dynamic plant***. Proc. IEEE 121. vol.12, 1973.

[2] 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.

[3] ***Mathworks - Fuzzy c-means clustering***. Disponível em: `<https://www.mathworks.com/help/fuzzy/fcm.html>`. Acesso em 22 de maio de 2017.