



Universidade Federal de Lavras
Departamento de Engenharia

- PSI 531 - Sistemas Fuzzy

Prof. Daniel Leite

E-mail: daniel.leite@deg.ufla.br

2/2017

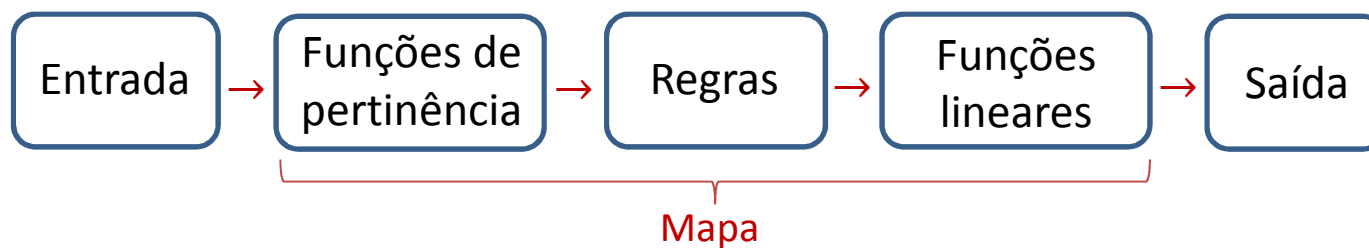
Sumário

- ANFIS recapitulação
- Interface de edição
- Projeto por linha de comando

Adaptive Neuro-Fuzzy Inference System

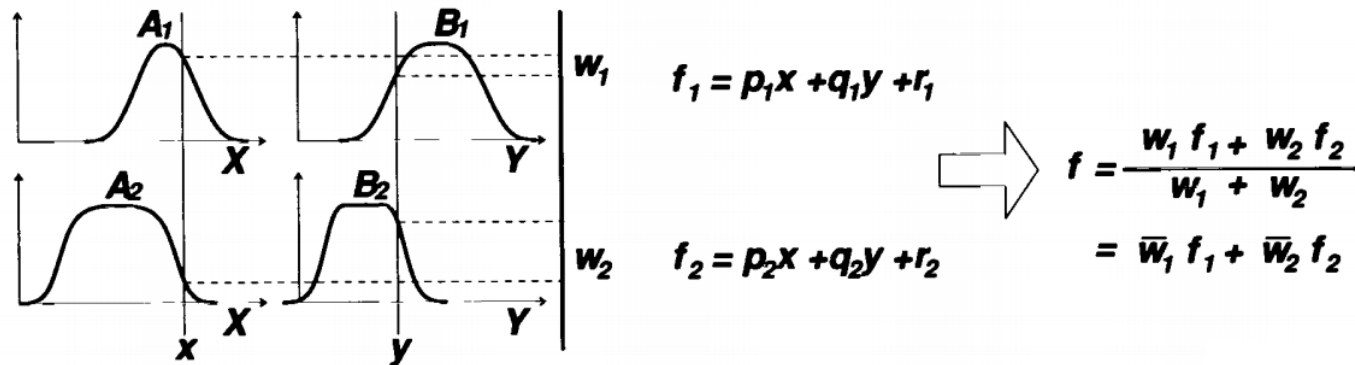
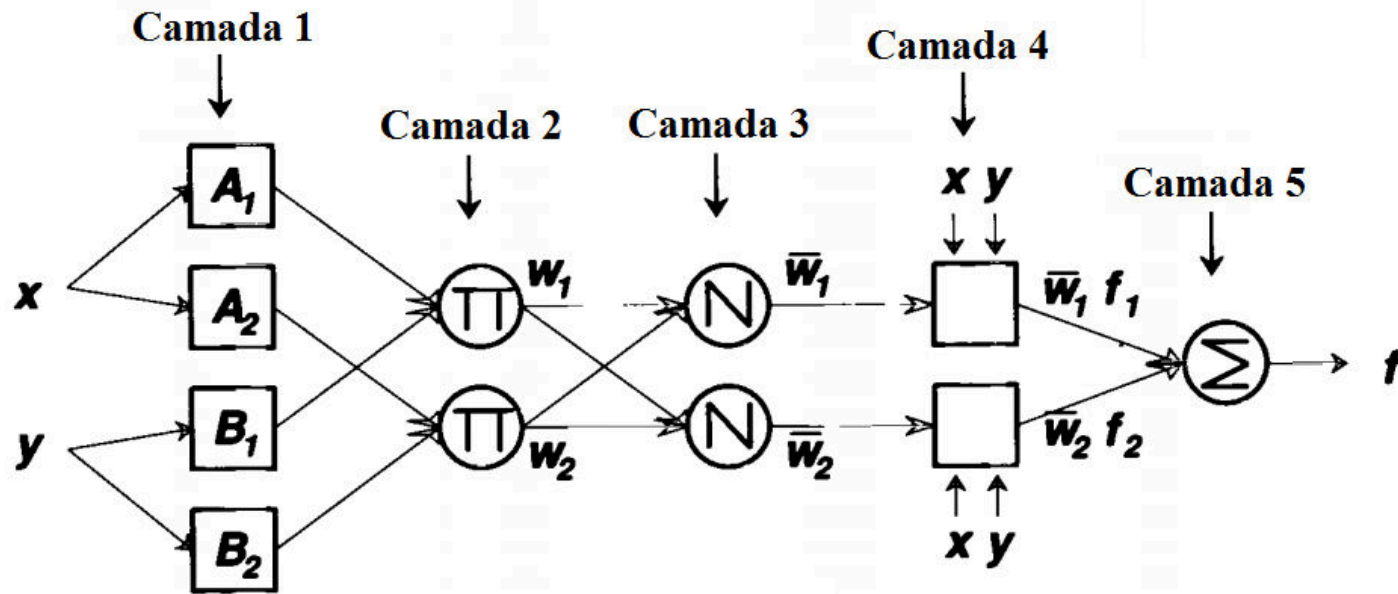
ANFIS Características

- Estrutura de rede similar a de uma rede neural (modelo paramétrico de um processo ou fenômeno)
- Regras e funções de pertinência são ajustados a partir de dados de entrada e saída (supervisão)



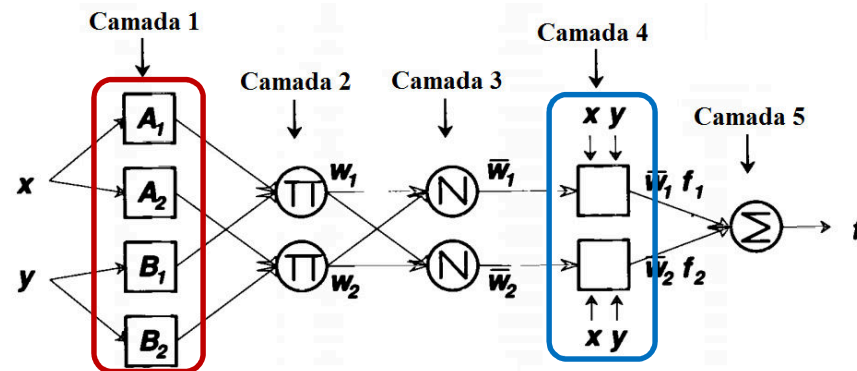
- Combina algoritmos de gradiente e quadrados mínimos

Arquitetura e raciocínio



Aprendizado

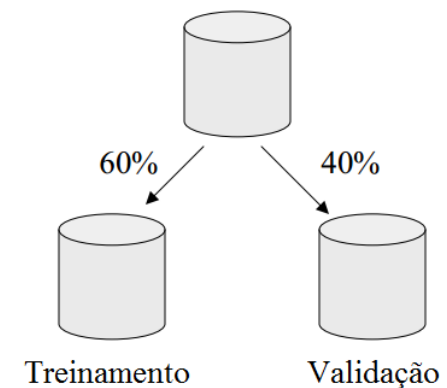
- Atualização 'desacoplada' de parâmetros antecedentes e consequentes
- Propagação adiante: adaptação dos parâmetros consequentes (lineares) via algoritmo de quadrados mínimos
- Retropropagação: adaptação dos parâmetros antecedentes (não-lineares) via algoritmo de gradiente



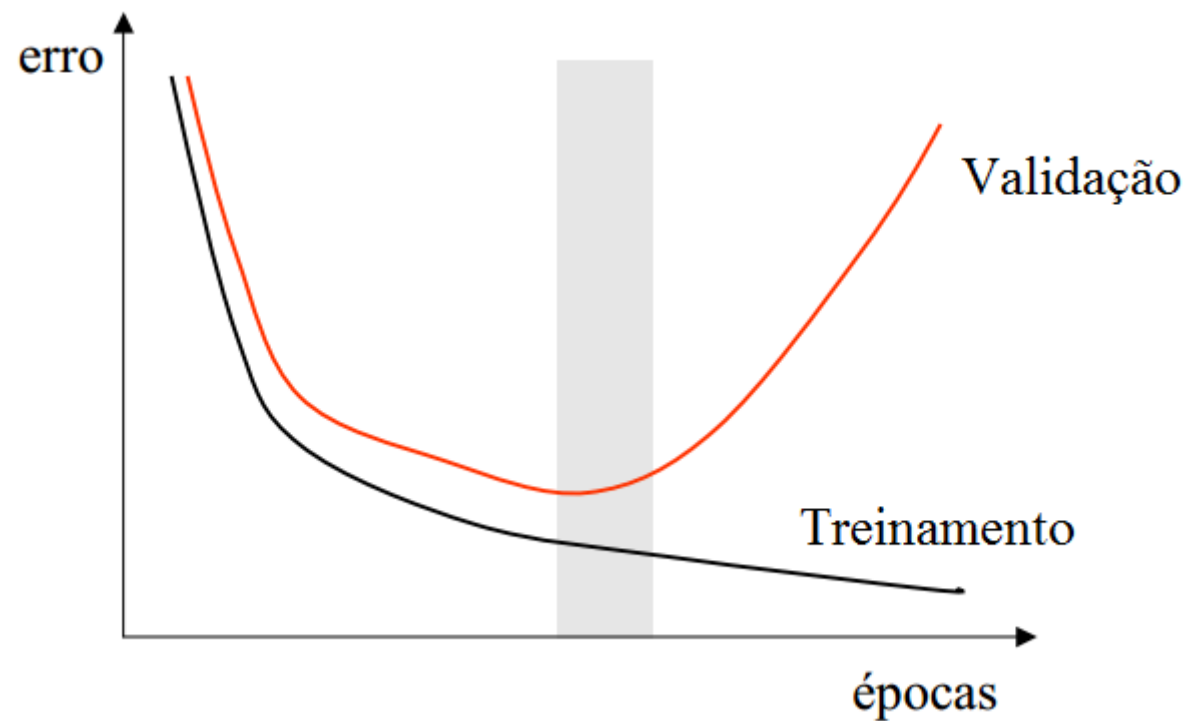
Validação de modelo

- Dados não utilizados em treino são úteis para verificar o quão bem novos valores de saída são previstos
- Dados de teste (ou de validação) permitem avaliar a capacidade de **generalização** do modelo
- Essencialmente, após um certo momento do treinamento, o modelo passa a apresentar *overfitting**
- Validação elimina/reduz efeito *overfitting*

* *Overfitting*: erro para dados de validação aumenta enquanto erro para dados de treino reduz



Validação cruzada



ANFIS Editor

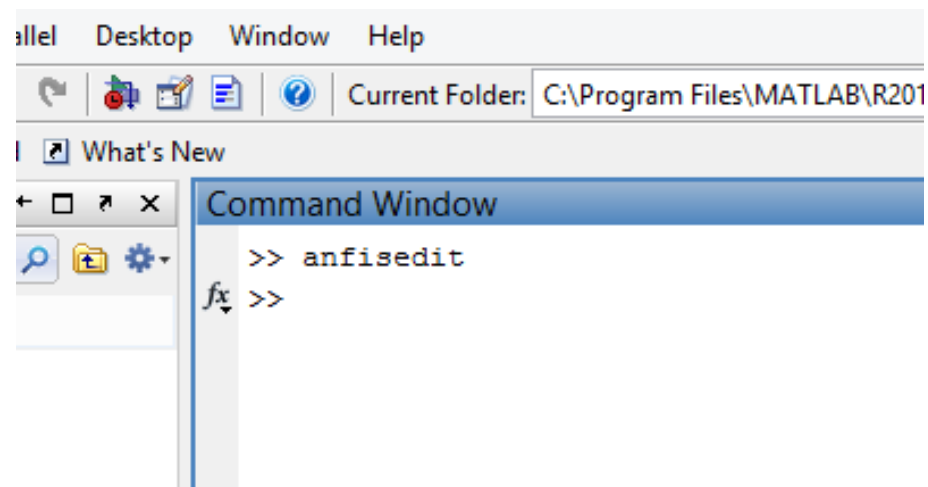
- Interface gráfica em Matlab para modelagem fuzzy tipo Takagi-Sugeno
- Acessível por editor GUI ou por linha de comando
- O editor gera código automaticamente

Limitações do editor

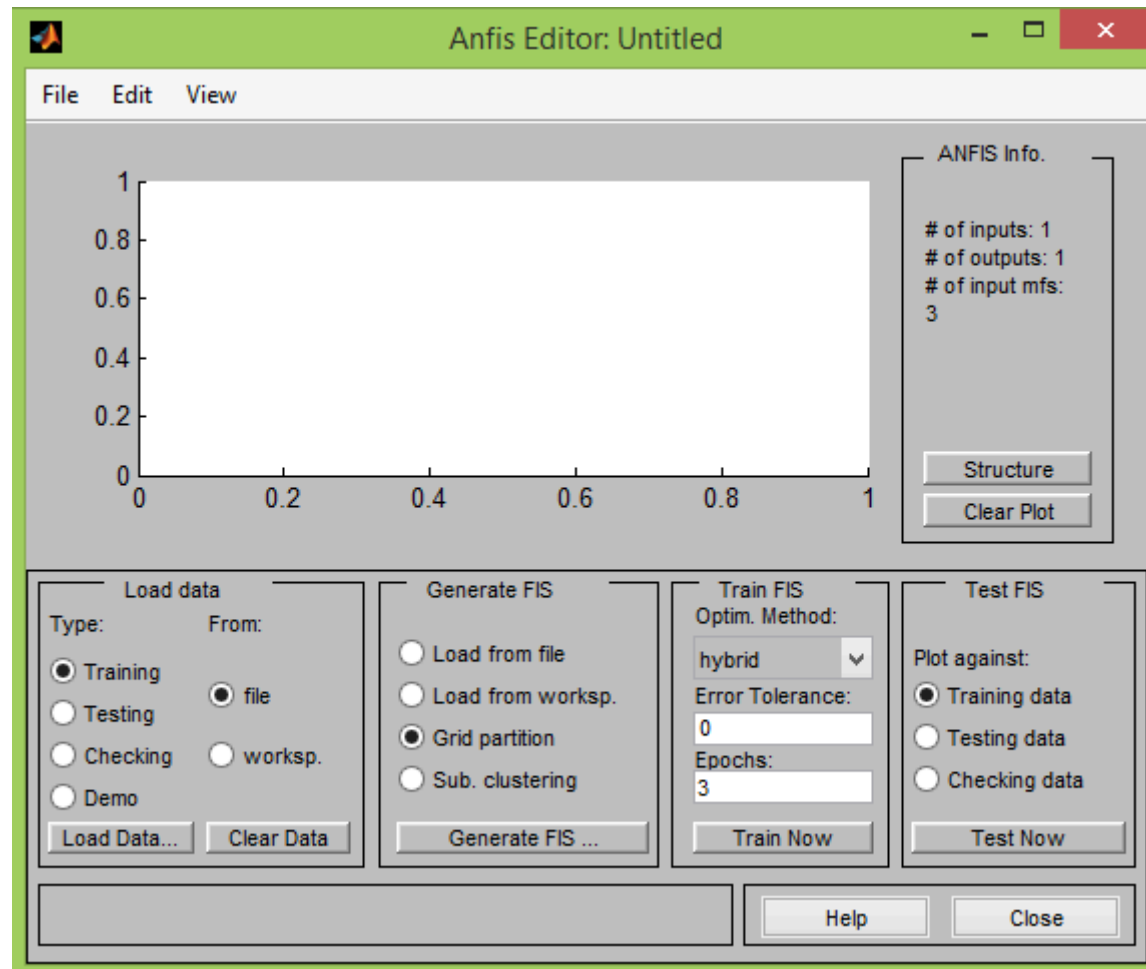
- Modelo fuzzy Takagi-Sugeno de ordem 0 ou 1
- Saída única, obtida por defuzzificação 'Média Ponderada'
- Funções de saída devem ser todas do mesmo tipo, i.e., lineares ou constantes
- Não há compartilhamento de regras, i.e., o número de funções de saída é igual ao número de regras
- O peso de todas as regras é '1'
- Customizações não são aceitas

Inicializando o editor

- Digitar anfisedit na janela de comando



ANFIS Editor



Carregando dados

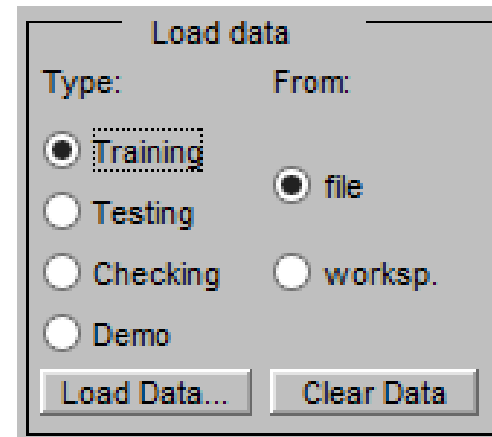
- Carregar base de dados de treinamento (**círculo**), teste (**diamante**) e/ou verificação (**cruz**)
- Dados devem estar na forma de matriz, onde as colunas são as variáveis. A última coluna é a saída

Especificar o tipo dos dados

Selecionar a origem: file/workspace

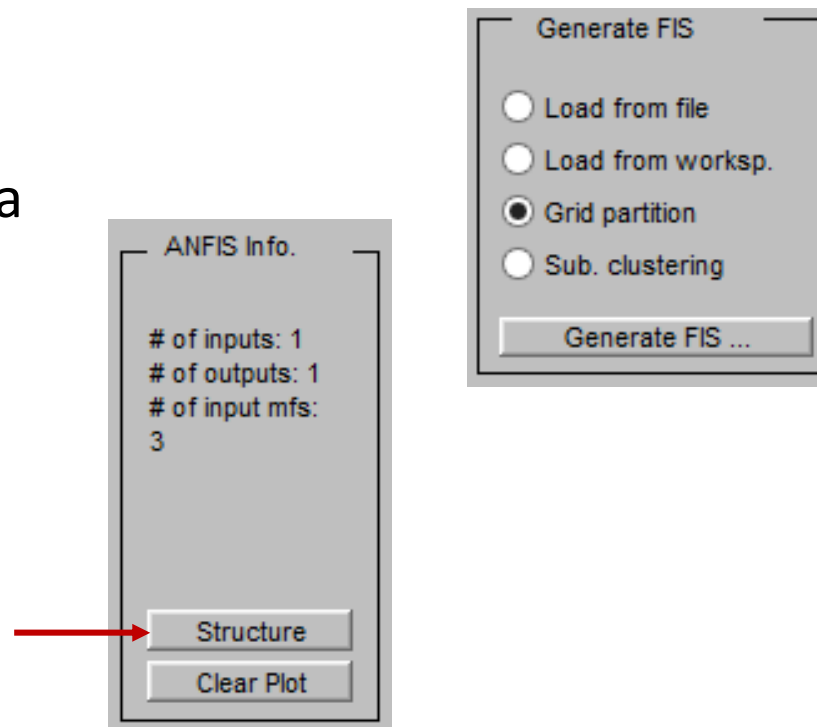
Carregar: *load data*

Apagar: *clear data*



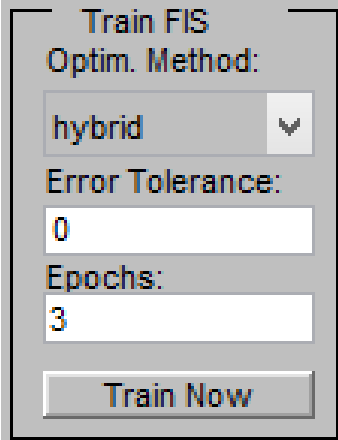
Gerando/Carregando FIS inicial

- Opções:
 - Carregar FIS Takagi-Sugeno a partir de arquivo ou *workspace*
 - Gerar modelo FIS inicial através de
 - Partição em grid
 - Subtractive clustering
- Para visualizar a estrutura do modelo FIS: *Structure*



Aprendizagem/Treinamento

- Escolher *Optimization Method*
 - *Hybrid* (Mínimos Quadrados e Gradiente Descendente)
 - *Backpropagation* (Gradiente Descendente)
- Escolher critérios de parada
 - Número máximo de épocas
 - Tolerância de erro
- Clicar *Train Now* para adaptar as funções de pertinência e visualizar a convergência do erro



Train FIS

Optim. Method:

hybrid ▼

Error Tolerance:

0

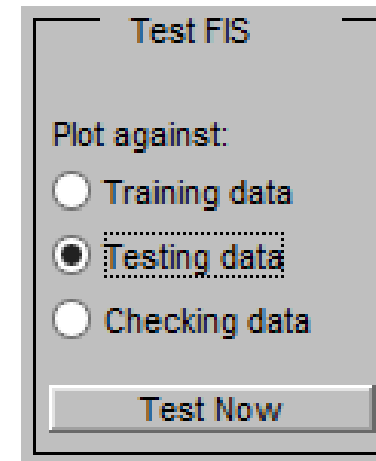
Epochs:

3

Train Now

Validando o FIS

- A base de **dados de validação ou teste*** deve estar carregada
- Clicar *Test Now* para plotar e comparar, graficamente, os dados de validação ou teste contra a saída do FIS (em vermelho)



* Dados representativos, mas suficientemente distintos daqueles usados no treinamento

Exemplo 1

- Carregar dados de treino e validação no *workspace*

```
>> load fuzex1trnData.dat
```

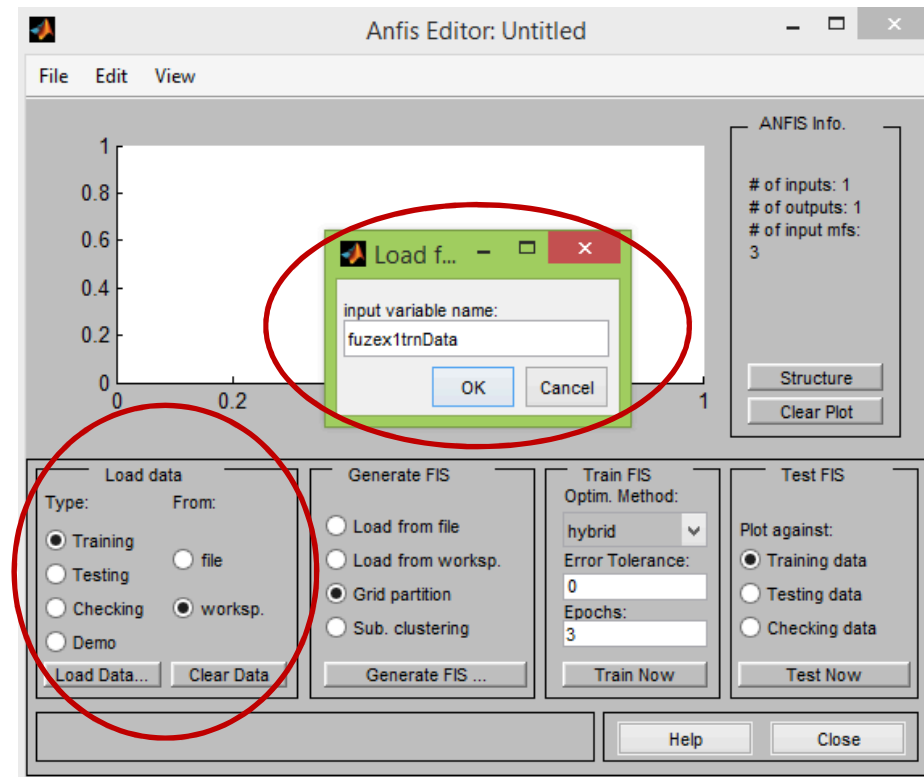
```
>> load fuzex1chkData.dat
```

- Abrir GUI Editor

```
>> anfisedit
```

- Em '*Load data*' selecionar:
Training e *worksp.*

'input variable name':
fuzex1trnData



- 25 dados de treinamento (círculos)
- Queremos um FIS tipo T-S para aproximar a função que descreve o comportamento do processo que gerou os dados



- Em '*Load data*' selecionar:

Checking e worksp.

'input variable name':

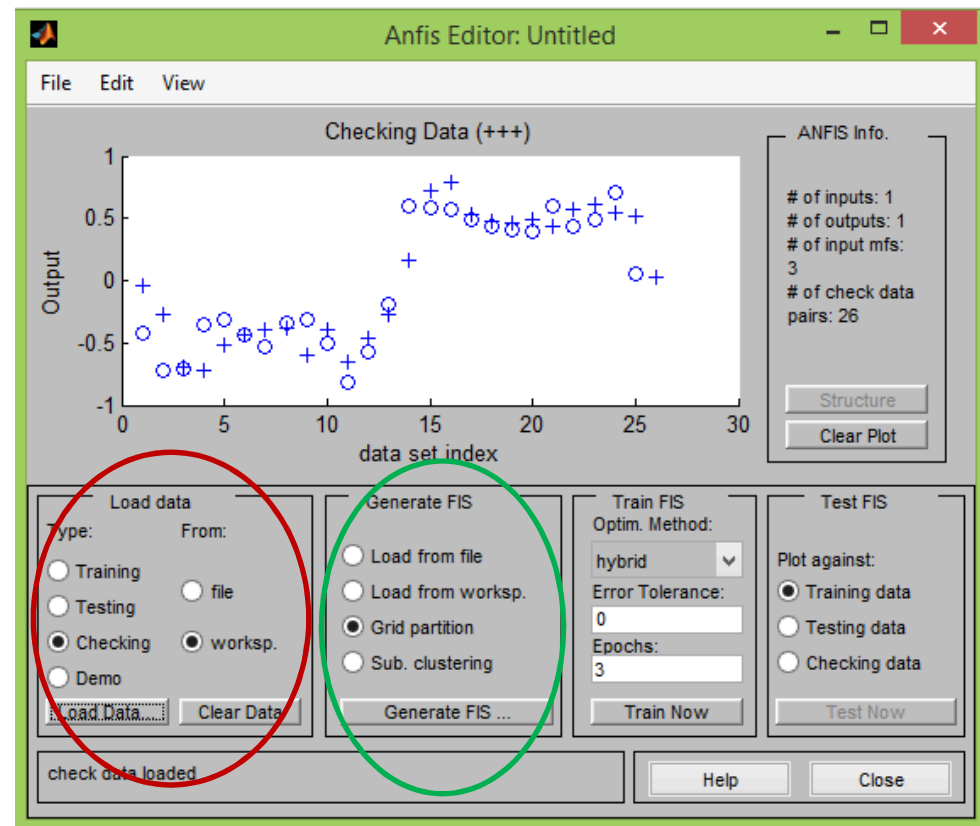
fuzex1chkData

- 26 dados de validação
(sinal +)

- Próximo passo:

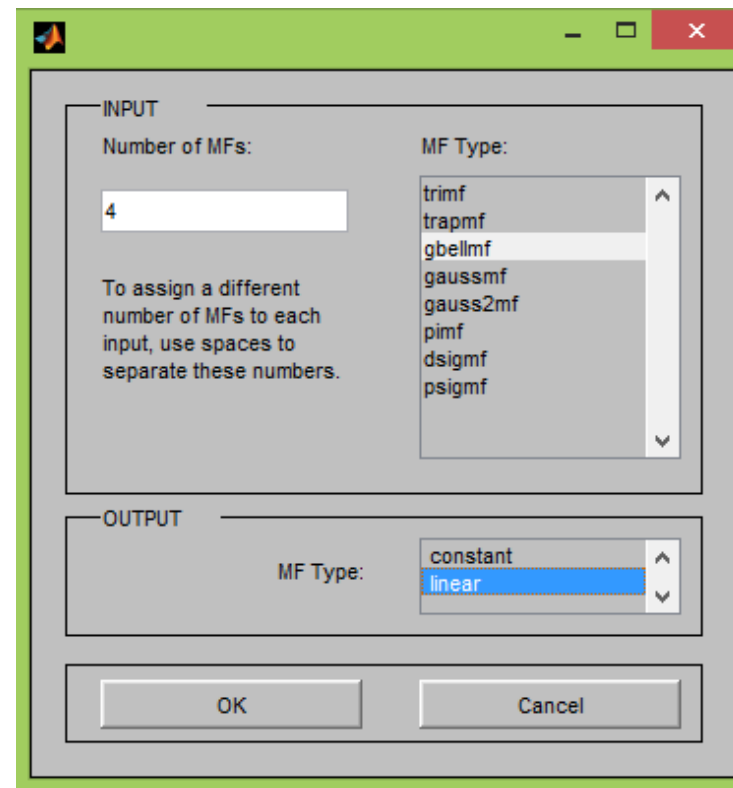
Carregar FIS inicial

ou gerar FIS

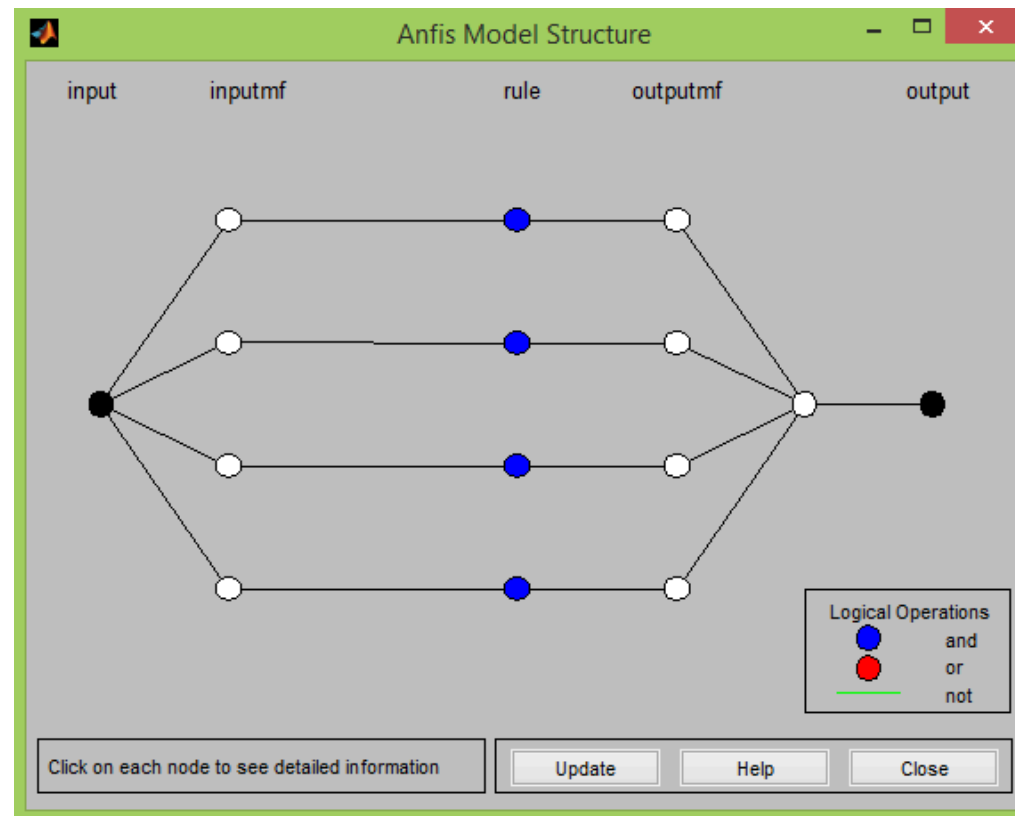


- Em '*Generate FIS*' escolher:
Grid partition (usa *Fuzzy C-Means Clustering*) → '*Generate FIS*'
- Preencher na janela aberta:
Input (4 MFs):
MF type: gbellmf (Gaussiana)
Output:
MF type: linear (T-S ordem 1)

-
- * Este FIS pode ser implementado em linha de comando usando:
- *genfis1* (*grid partitioning*)
 - *genfis2* (*subtractive clustering*)



- Clicar em '*Structure*' para visualizar o FIS (ANFIS) gerado



- Treinamento FIS

Selecionar:

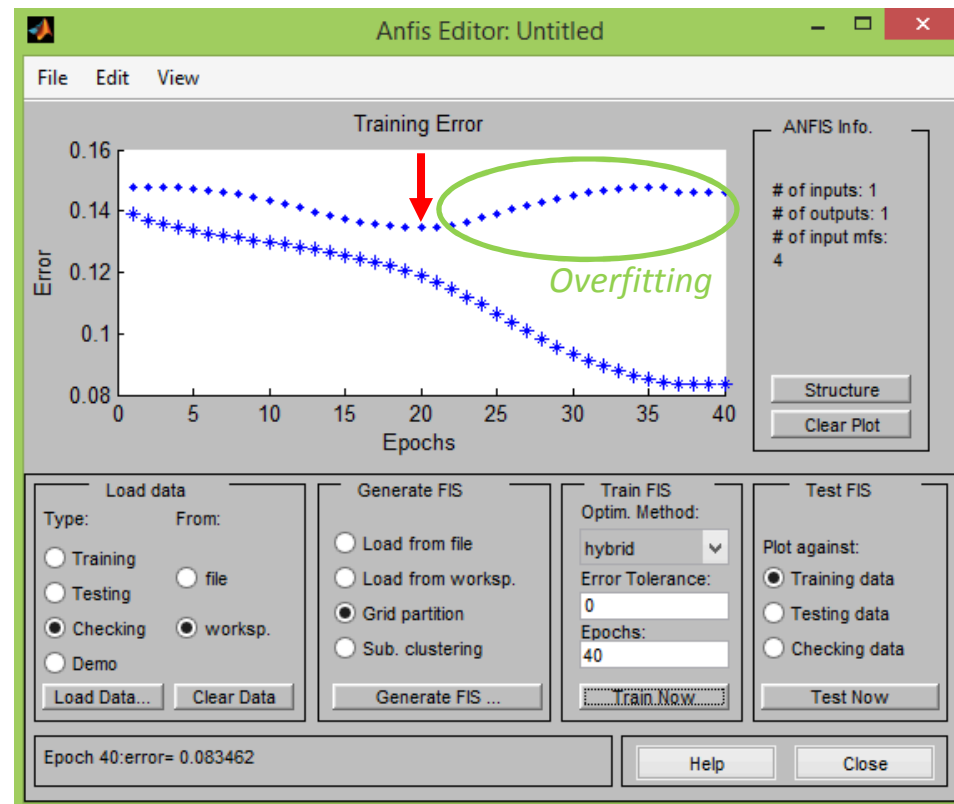
Optim. Method: Hybrid

Error tolerance: 0

Epochs: 40

Clicar: *Train Now*

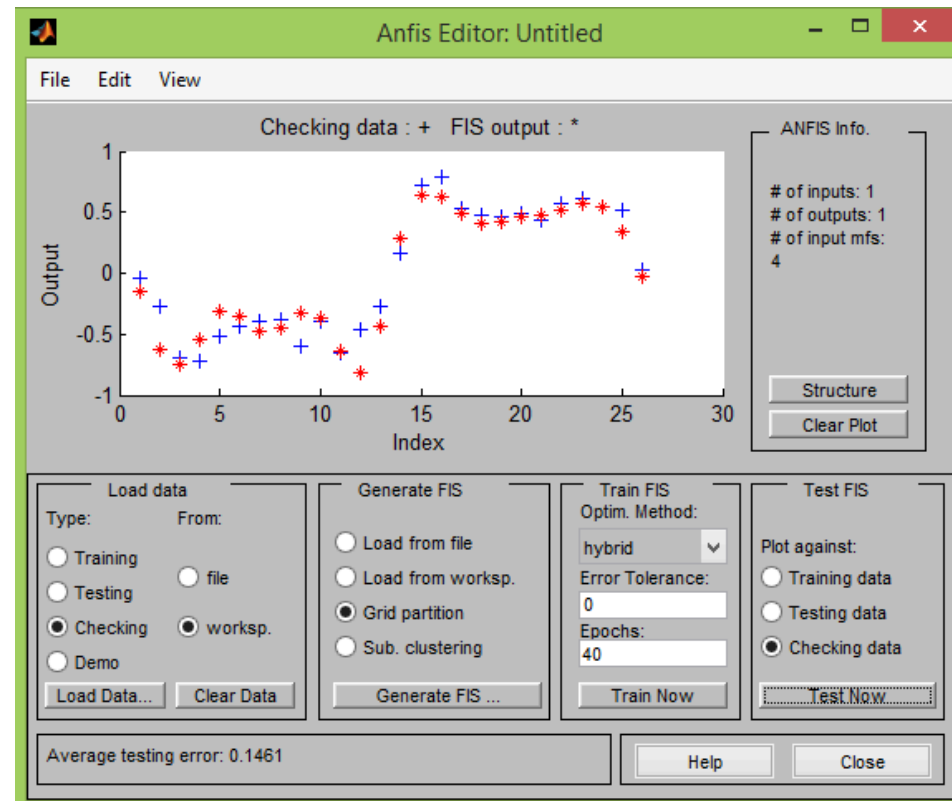
* Os parâmetros do FIS são tomados no ponto de mínimo erro para dados de validação



- Teste FIS

Escolher *Checking data*
em *Test FIS*
Clique: *Test Now*

* A aproximação parece
satisfatória



Exemplo 2: linha de comando

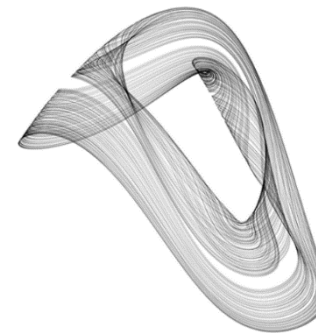
- Previsão da série temporal caótica Mackey-Glass

$$\dot{x} = \frac{0,2x(t-\tau)}{1+x^{10}(t-\tau)} - 0,1x(t) \quad : \text{eq. diferencial com atraso de tempo}$$

Trajetória sensível a condição inicial

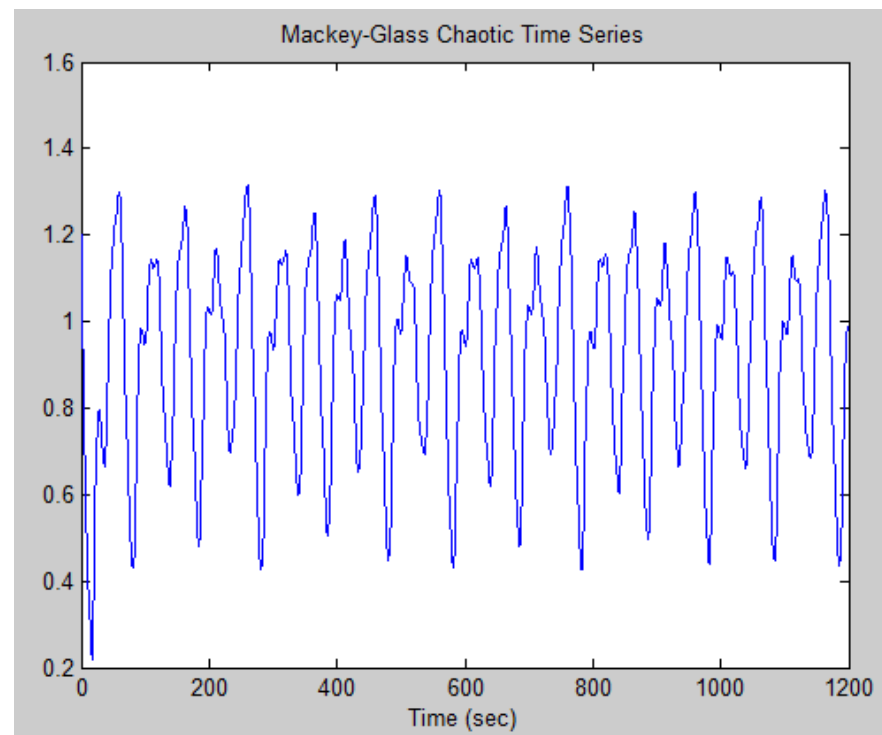
Algoritmo Runge-Kutta de 4ª ordem: útil para obter solução numérica

O resultado (série) está salvo em: *mgdata.dat*



- Assume $x(0) = 1.2$, $\tau = 17$, e $x(t) = 0$ para $t < 0$

```
>> load mgdata.dat  
>> time = mgdata(:, 1);  
>> x = mgdata(:, 2);  
>> figure(1), plot(time, x);  
>> title('Mackey-Glass Time Series')  
>> xlabel('Time (sec)')
```



- Assume-se para treinamento de ANFIS

Entradas: $w(t) = [x(t - 18) \ x(t - 12) \ x(t - 6) \ x(t)]$

Saída: $s(t) = x(t + 6)$

Tempo: $118 \leq t \leq 1117$

São 1000 vetores de entrada-saída; sendo os primeiros 500 usados para treinamento ANFIS e os demais para validação do FIS identificado

```
for t = 118:1117
```

```
    Data(t-117,:) = [x(t-18) x(t-12) x(t-6) x(t) x(t+6)];
```

```
end
```

```
trnData = Data(1:500, :);
```

```
chkData = Data(501:end, :);
```

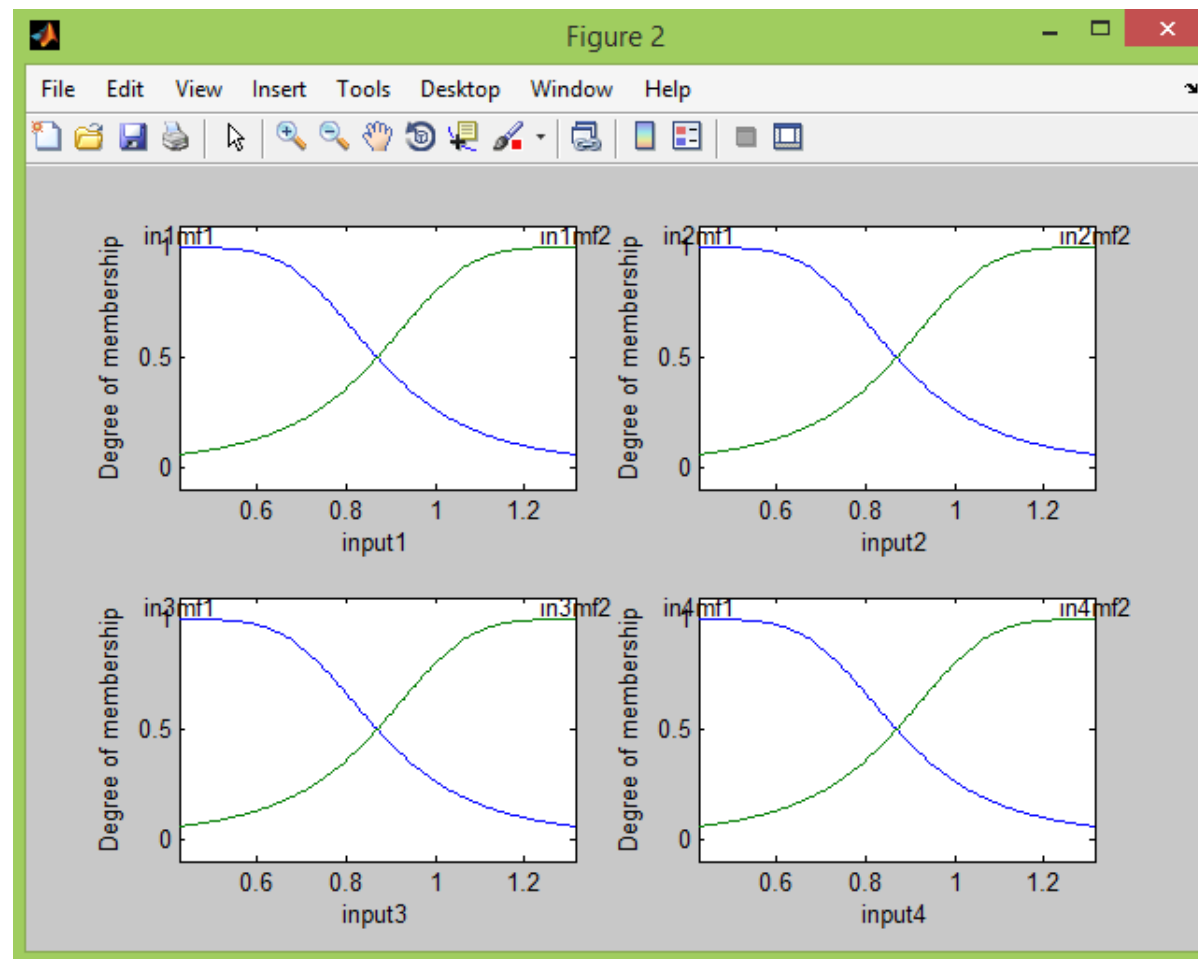
- Gerar FIS inicial (partição em grid via Fuzzy C-Means)

```
>> fismat = genfis1(trnData);    %Valores default para MFs  
  
% Duas MFs Sino Generalizada geradas para cada entrada; 8 MFs total  
% Estrutura FIS: 16 regras, 104 parâmetros  
% Ratio data/parameters (500/104); em torno de 5
```

- Plotar funções de pertinência

```
>> figure(2)  
>> subplot(2,2,1), plotmf(fismat, 'input', 1)  
>> subplot(2,2,2), plotmf(fismat, 'input', 2)  
>> subplot(2,2,3), plotmf(fismat, 'input', 3)  
>> subplot(2,2,4), plotmf(fismat, 'input', 4)
```

- Funções de pertinência **iniciais**



- Treinamento FIS

```
>> [fismat1,error1,ss,fismat2,error2] = anfis(trnData,fismat,[],[],chkData);
```

% fismat1: associado ao mínimo erro de treinamento

% fismat2: associado ao mínimo erro de validação

- Plotar funções de pertinência

```
>> figure(3)
```

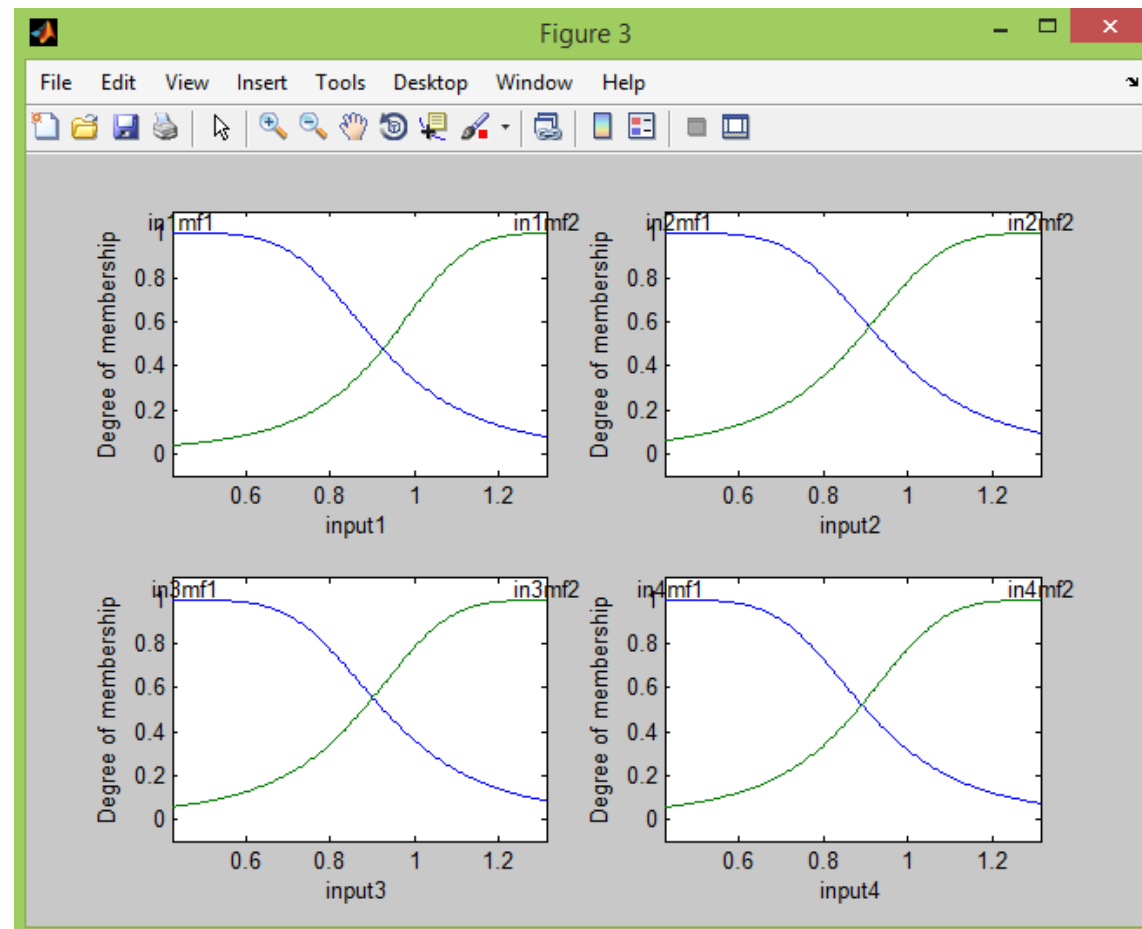
```
>> subplot(2,2,1), plotmf(fismat2, 'input', 1)
```

```
>> subplot(2,2,2), plotmf(fismat2, 'input', 2)
```

```
>> subplot(2,2,3), plotmf(fismat2, 'input', 3)
```

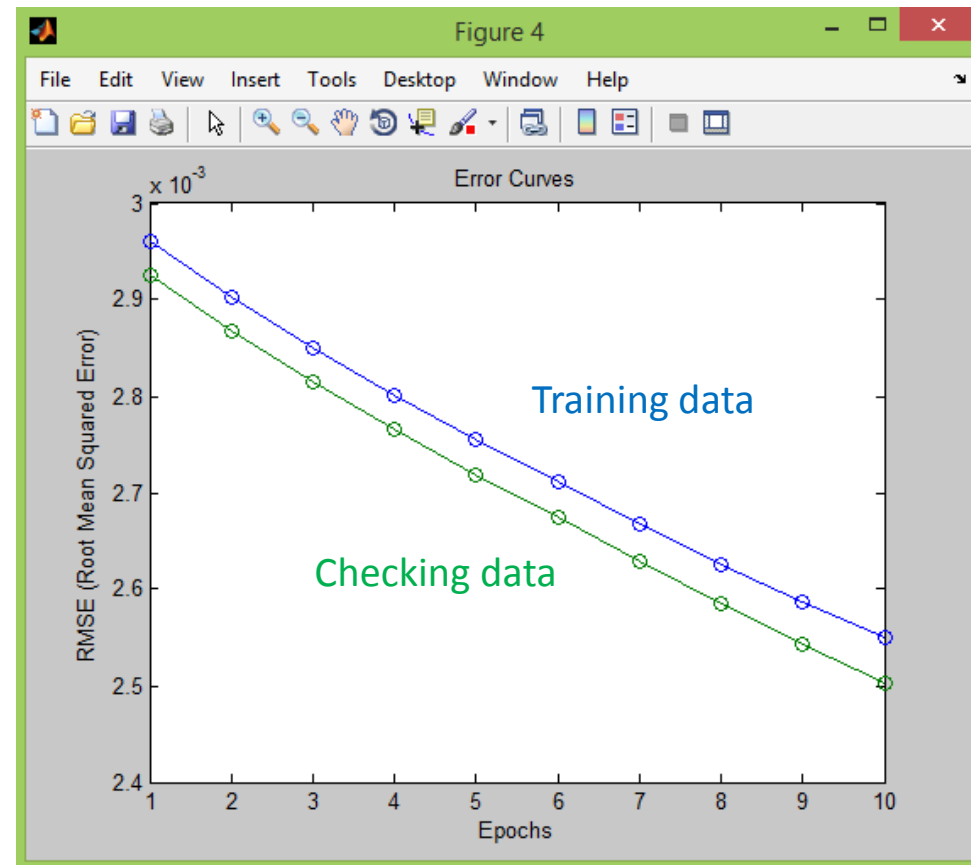
```
>> subplot(2,2,4), plotmf(fismat2, 'input', 4)
```

- Funções de pertinência **finais**



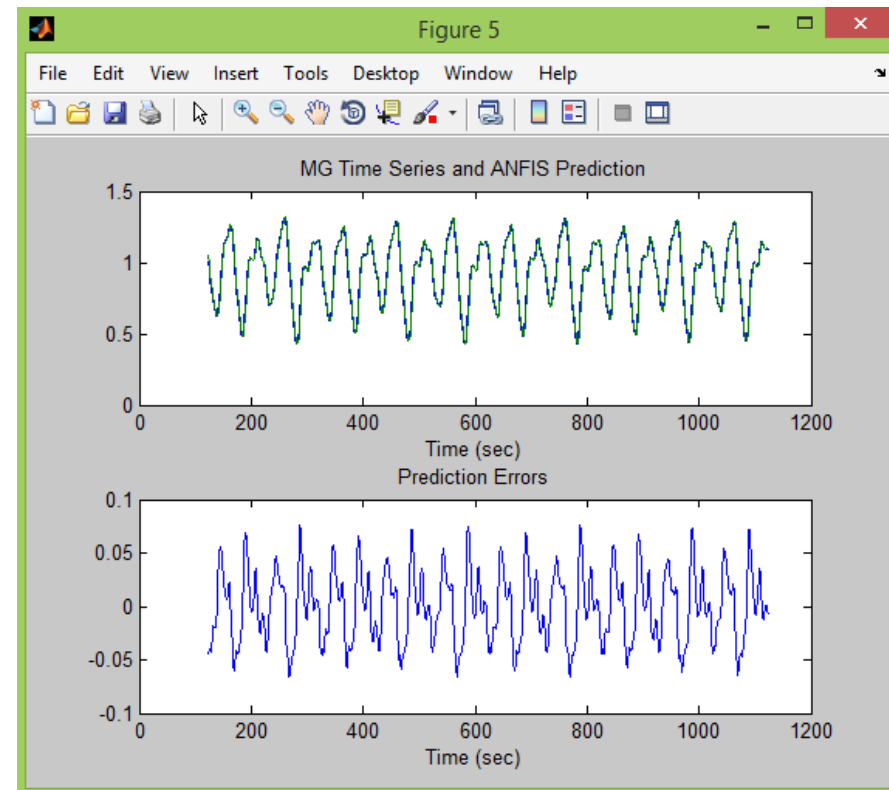
- Raiz do erro quadrado médio

```
>> figure(4)
>> plot([error1 error2]);
>> hold on; plot([error1 error2], 'o');
>> xlabel('Epochs');
>> ylabel('RMSE');
>> title('Error Curves');
```



- Comparação série Mackey-Glass original × previsão FIS

```
>> figure(5)
>> anfis_output = evalfis([trnData(:,1:4);
    chkData(:,1:4)], fismat2);
>> index = 125:1124;
>> subplot(2,1,1), plot(time(index),
    [x(index) anfis_output]);
>> xlabel('Time (sec)');
>> title('MG Series and ANFIS Prediction');
>> subplot(2,1,2), plot(time(index),
    x(index) - anfis_output);
>> xlabel('Time (sec)');
>> title('Prediction Errors');
```



Procedimento geral

- Caso 1 - Somente dados de treino disponíveis

```
>> [FIS,ERROR] = anfis(TRNDATA)
```

TRNDATA: dados treino - matriz N colunas (entradas) e uma coluna (saída)

ERROR: matriz raiz do erro quadrado médio

FIS: Sistema fuzzy final

* anfis usa GENFIS1 para criar FIS inicial com partição em grid

- Caso 2 - Dados de treino e FIS inicial disponíveis

```
>> [FIS,ERROR] = anfis(TRNDATA,INITFIS)
```

INITFIS: Sistema fuzzy inicial

FIS: Sistema fuzzy final

- Caso 3 - Dados de treino e FIS inicial disponíveis, opções ANFIS

```
>> [FIS,ERROR,STEPSIZE] = anfis(TRNDATA,INITFIS,TRNOPT,DISOPT,[],OPTMETHOD)
```

TRNOPT: especifica opções de treinamento

TRNOPT(1): número de épocas (default: 10)

TRNOPT(2): erro treinamento objetivo (default: 0)

TRNOPT(3): tamanho do passo inicial (default: 0.01)

TRNOPT(4): taxa redução passo (default: 0.9)

TRNOPT(5): taxa aumento passo (default: 1.1)

DISOPT: mostra evolução de variáveis (1: liga; 0: oculta)

DISOPT(1): informações gerais do anfis (default: 1)

DISOPT(2): erro (default: 1)

DISOPT(3): tamanho do passo a cada atualização (default: 1)

DISOPT(4): resultados finais (default: 1)

OPTMETHOD: método de otimização (1: hybrid; 0: backpropagation)

- Caso 4 - Dados de treino e FIS inicial disponíveis, opções ANFIS

```
>> [FIS,ERROR,STEPSIZE,CHKFIS,CHKERROR] = ...  
    anfis(TRNDATA,INITFIS,TRNOPT,DISPOPT,CHKDATA,OPTMETHOD)
```

CHKDATA: dados validação - matriz N colunas (entradas) e uma coluna (saída)

CHKFIS: FIS ótimo - erro para dados de validação atinge mínimo

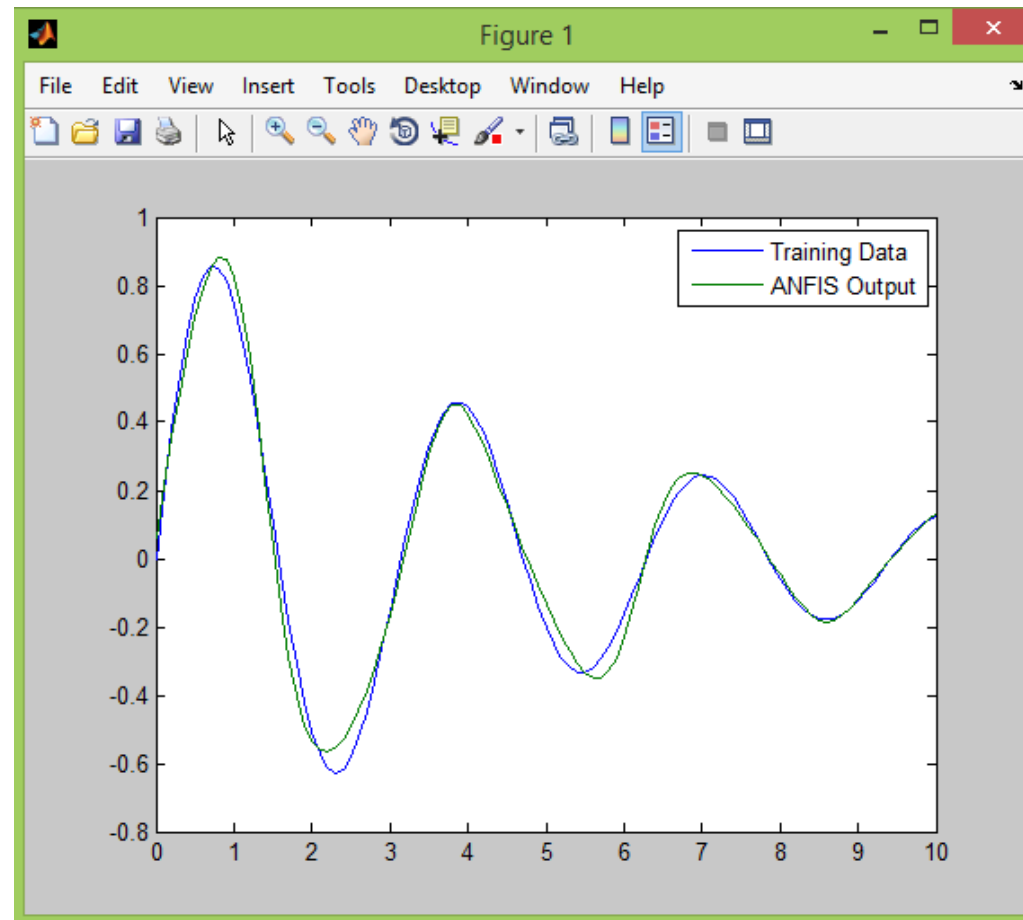
CHKERROR: matriz raiz do erro quadrado médio

* *Overfitting*: erro para dados de validação começa a aumentar enquanto erro para dados treino ainda reduz

Exemplo (caso 3)

```
>> x = (0:0.1:10)';  
>> y = sin(2*x)./exp(x/5);  
>> trnData = [x y];  
>> numMFs = 5;  
>> mfType = 'gbellmf';  
>> epoch_n = 20;  
>> in_fis = genfis1(trnData,numMFs,mfType);  
>> out_fis = anfis(trnData,in_fis,20);  
>> plot(x,y,x,evalfis(x,out_fis));  
>> legend('Training Data','ANFIS Output');  
>> showfis(out_fis)
```

- Aproximação FIS dados treino



- Parâmetros do FIS gerado (`showfis(out_fis)`)

1. Name	<i>anfis</i>
2. Type	<i>sugeno</i>
3. Inputs/Outputs	<i>[1 1]</i>
4. NumInputMFs	<i>5</i>
5. NumOutputMFs	<i>5</i>
6. NumRules	<i>5</i>
7. AndMethod	<i>prod</i>
8. OrMethod	<i>max</i>
9. ImpMethod	<i>prod</i>
10. AggMethod	<i>max</i>
11. DefuzzMethod	<i>wtaver</i>
14. InRange	<i>[0 10]</i>
15. OutRange	<i>[-0.6273 0.8567]</i>
26. InMFTypes	<i>gbellmf</i>

31. OutMFTypes	<i>linear</i>
36. InMFParams	<i>[1.397 2.005 0.1369 0]</i>
37.	<i>[1.257 1.967 2.476 0]</i>
38.	<i>[1.231 1.995 4.98 0]</i>
39.	<i>[1.31 1.999 7.447 0]</i>
40.	<i>[1.252 1.999 9.999 0]</i>
41. OutMFParams	<i>[1.979 0.3108 0 0]</i>
42.	<i>[1.115 -4.011 0 0]</i>
43.	<i>[-0.61 2.636 0 0]</i>
44.	<i>[-0.4502 3.594 0 0]</i>
45.	<i>[0.1557 -1.357 0 0]</i>

```

1. If (input1 is in1mf1) then (output is out1mf1) (1)
2. If (input1 is in1mf2) then (output is out1mf2) (1)
3. If (input1 is in1mf3) then (output is out1mf3) (1)
4. If (input1 is in1mf4) then (output is out1mf4) (1)
5. If (input1 is in1mf5) then (output is out1mf5) (1)

```

Projeto 3 – 10 pontos

- Escolher dois problemas de regressão (P1 e P2) no site:

<http://archive.ics.uci.edu/ml/datasets.html>

- Separar base de dados de validação e de teste
- Implementar ANFIS via linha de comando para P1
- Implementar ANFIS via interface para P2
- Entregar arquivo.pdf contendo breve explicação dos problemas, considerações, modelos e gráficos

Observação

Este material refere-se às notas de aula do curso PSI-531, Sistemas Fuzzy, do Programa de Pós-Graduação em Engenharia de Sistemas e Automação da UFLA. Ele não substitui as referências recomendadas. Este material não pode ser reproduzido sem autorização dos autores. Quando autorizado, seu uso é exclusivo para ensino em instituições sem fins lucrativos.