THE COMPARISON BETWEEN ANFIS AND RBFN USING LS

Weicheng Guo

PURDUE
UNIVERSITY

PROBLEM DEFINITION

Use Adaptive Neural Fuzzy Inference System (ANFIS) and Radial Basis Function Network (RBFN) with Least Square (LS) method to fit the following nonlinear function:

$$y = \frac{1}{3}(x_1 + x_2)\sin(x_1 + x_2)e^{-\frac{1}{9}(x_1+x_2)} - \frac{1}{5}(x_2 + x_3)\sin(x_2 + x_3)e^{-\frac{1}{9}(x_2+x_3)}$$

$$x_1, x_2, x_3 \in [-3,3]$$

where x1, x2 and x3 are the inputs, y is the output.

PURDUE UNIVERSITY

ANFIS FUNCTION IN MATLAB

1. ANFIS

Syntax: fismat = genfis1(data,numMFs,inmftype,outmftype)

genfis1.m function generates a T-S type FIS structure used as initial conditions (initialization of the membership function parameters) for ANFIS training.

Inputs:

- data : the training data matrix, which must be entered with all but the last columns representing input data, and the last column representing the single output.
- numMFs : a vector whose coordinates specify the number of membership functions associated with each input.
- inmftype : a string array in which each row specifies the membership function type associated with each input.
- outmftype : a string that specifies the membership function type associated with the output. There can only be one output, because this is a T-S type system.

PURDUE UNIVERSITY

ANFIS FUNCTION IN MATLAB

Syntax: [fis,error,stepsize,chkFis,chkErr]
=anfis(trnData,initFIS,trnOpt,dispOpt,chkData)

anfis.m function uses a hybrid learning algorithm to tune the parameters of a T-S type fuzzy inference system (FIS).

Inputs:

- trnData : training data, specified as a matrix. For an FIS with N inputs, trnData has N+1 columns, where the first N columns contain input data and the final column contains output data.
- initFIS : FIS structure used to provide an initial set of membership functions for training. Typically, we use fismat generated by genfis1.
- trnOpt, dispOpt : training options and display options, we can use default settings.
- chkData : validation data used to prevent overfitting of the training data, specified as a matrix. This matrix is in the same format as trnData.

PURDUE UNIVERSITY

ANFIS FUNCTION IN MATLAB

Syntax: [fis,error,stepsize,chkFis,chkErr]
=anfis(trnData,initFIS,trnOpt,dispOpt,chkData)

Outputs:

- fis : FIS structure whose parameters are tuned using the training data, returned as a structure.
- error : root mean squared training data errors at each training epoch, returned as an array of scalars.
- stepsize : step sizes at each training epoch, returned as an array of scalars.
- chkFis : FIS structure that corresponds to the epoch at which chkErr is minimum.
- chkErr : root mean squared checking data errors at each training epoch, returned as an array of scalars.

PURDUE UNIVERSITY

STEP 1 : INITIALIZE ANFIS

1. Use GenerateTrainingData.m function to generate the training inputs x, training inputs d, checking inputs Cx and checking outputs Cd.

```
n=5000;                          % 5000 samples for training
m=1000;                          % 1000 samples for checking
InputRange=[-3 3;-3 3;-3 3];   % 3 inputs
[x,Cx,d,Cd]=GenerateTrainingData(n,m,InputRange);
```

2. Set initial parameters of ANFIS

```
td=[x' d'];              % training data for ANFIS
ckd=[Cx' Cd'];           % checking data for ANFIS
numMFs=9;                % choose maximum number of membership functions
mfType='gaussmf';    % choose gaussian curve membership functions
```


PURDUE UNIVERSITY

STEP 2 : PERFORM THE TRAINING

1. Use genfis1.m function to generate a three-input and one-output T-S type FIS structure with initial parameters.

2. Use anfis.m function to tune the parameters of membership functions in T-S type fuzzy inference system.

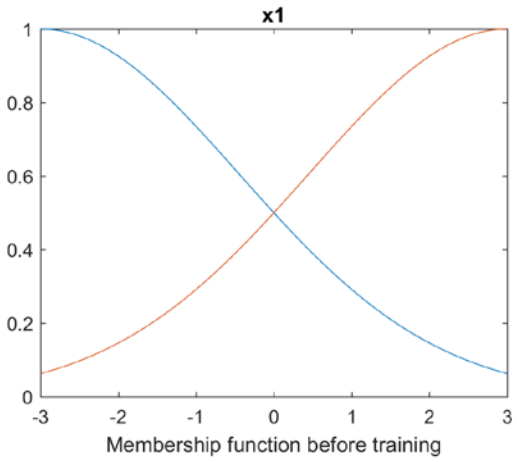
3. Calculate the error in ANFIS with checking data using the following criterion:

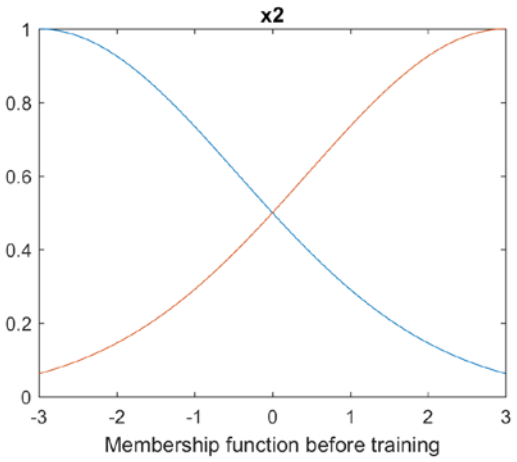
$$NDEI = \sqrt{\frac{\sum_{k=1}^{N}[d(k) - y(k)]^2}{\sum_{k=1}^{N}\left[d(k) - \bar{d}\right]^2}}$$

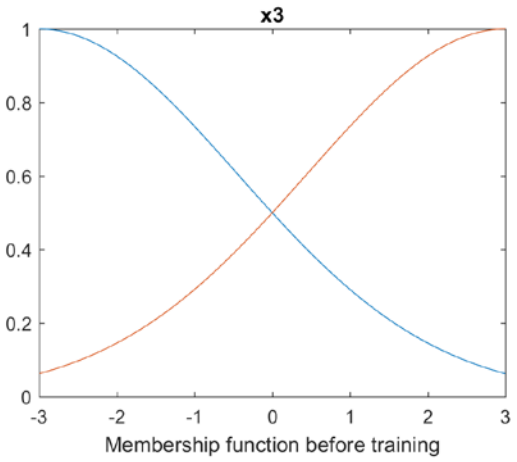
PURDUE UNIVERSITY

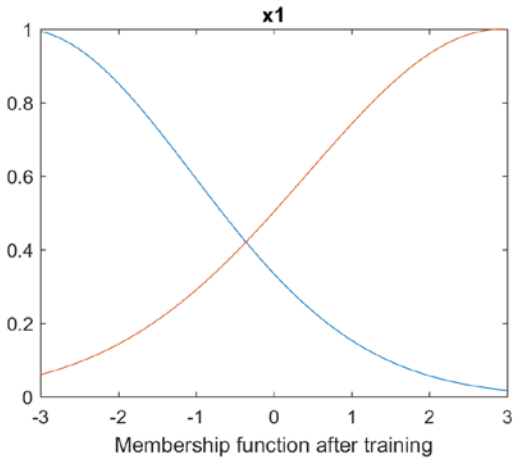
TRAINING RESULT

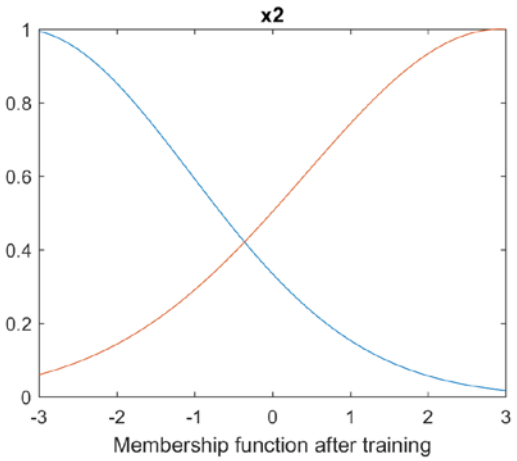
Number of MF = 2

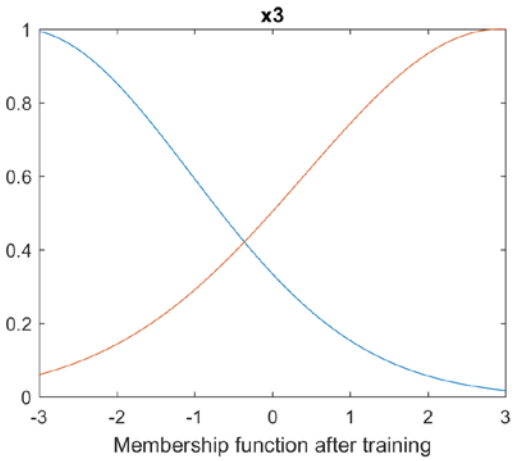








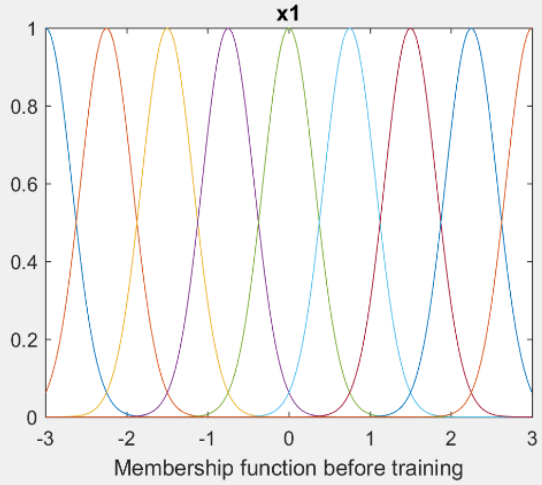


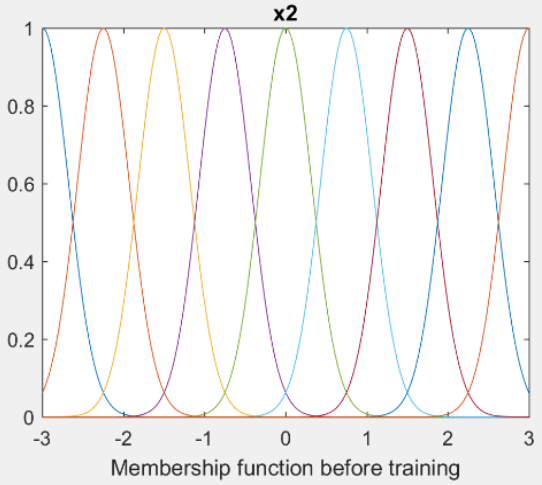


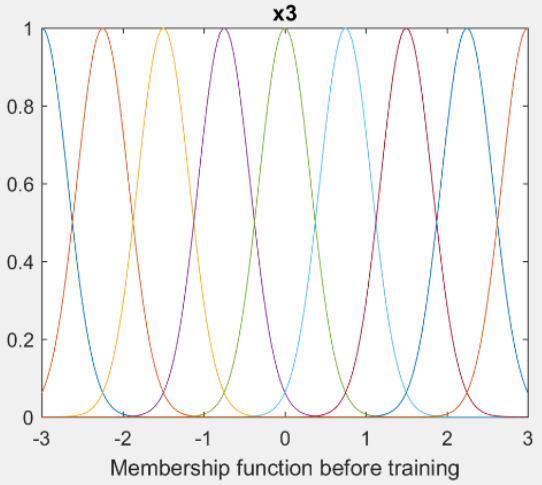
PURDUE UNIVERSITY

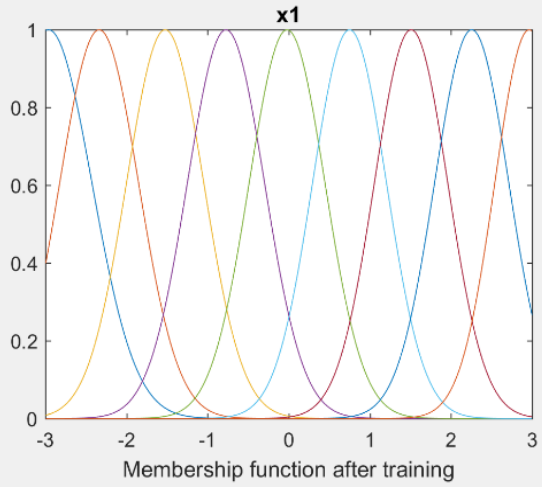
TESTING RESULT

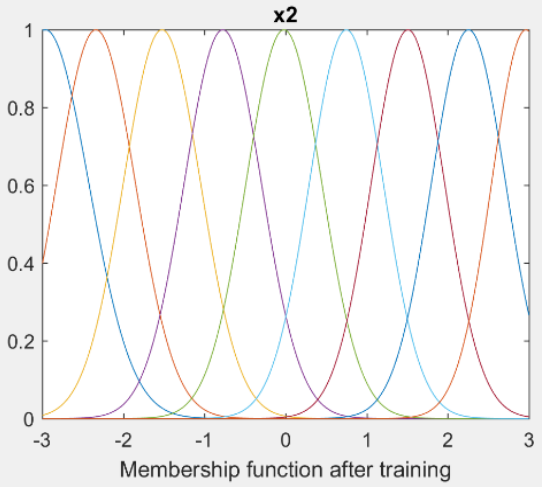
Number of MFs =9

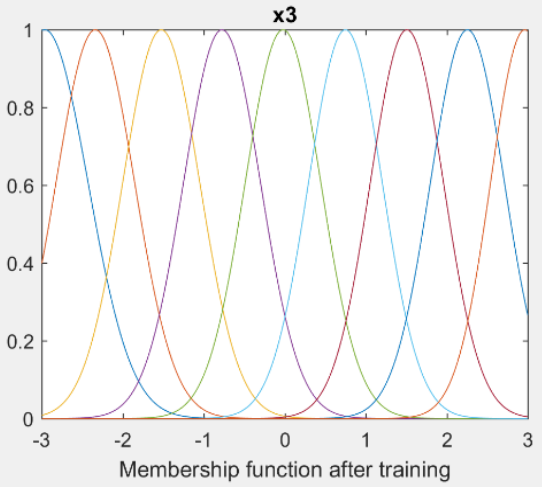








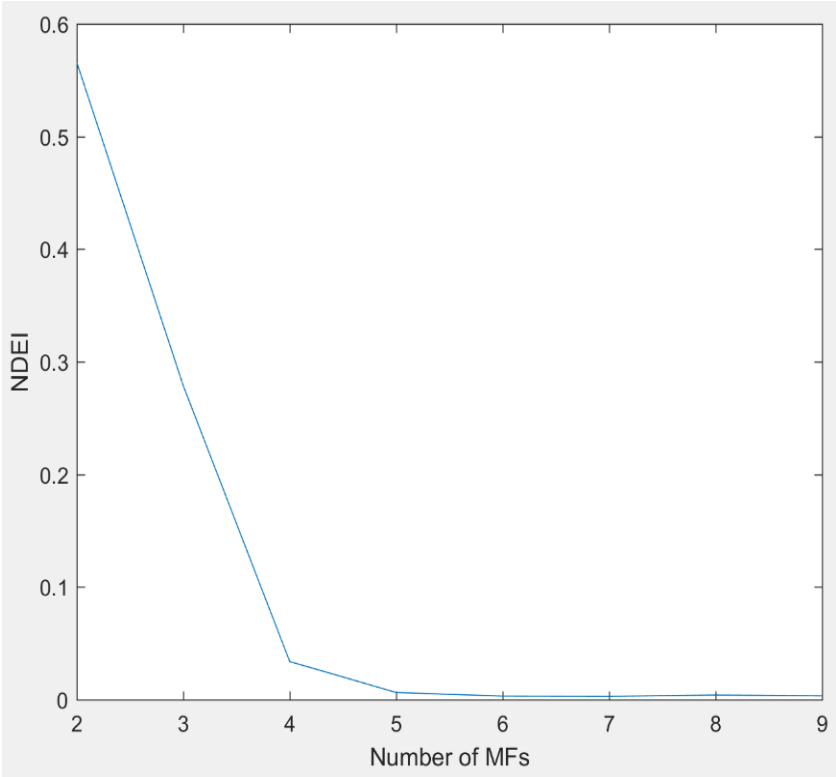


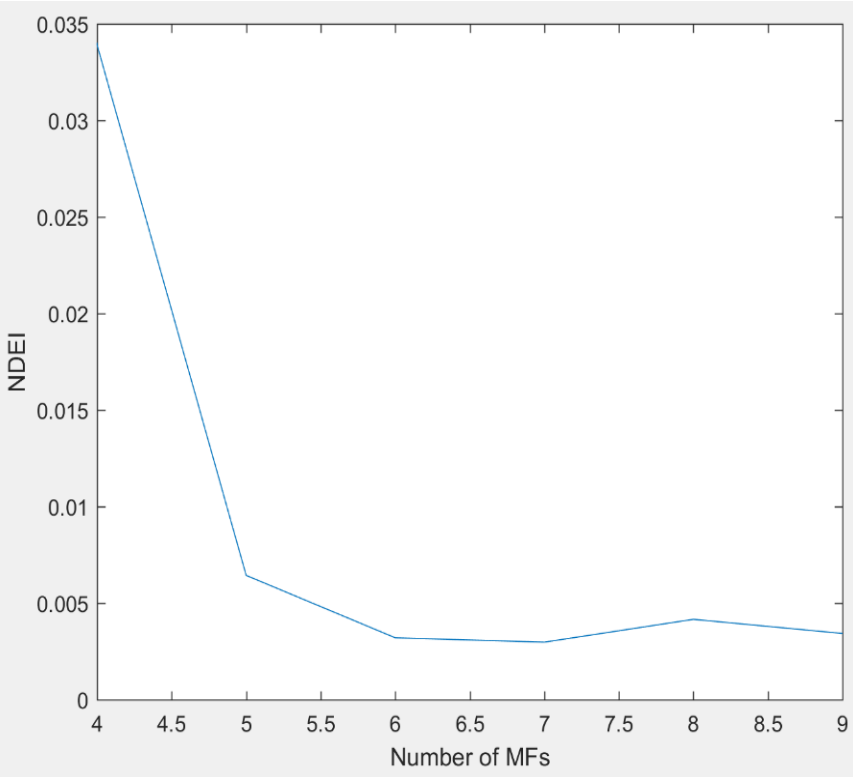


PURDUE UNIVERSITY

TESTING RESULT

ERROR REDUCTION





No. of MFs	2	3	4	5	6	7	8	9
NDEI	0.5658	0.2788	0.0339	0.0064	0.0032	0.0030	0.0042	0.0034

PURDUE UNIVERSITY

2. RBFN WITH LEAST SQUARE METHOD

Syntax: Z = dist(W,P)

dist.m function is Euclidean distance weight function. It applies weights to an input to get weighted inputs.

Inputs:

- W : S-by-R weight matrix
- P : R-by-Q matrix of Q input (column) vectors

PURDUE UNIVERSITY

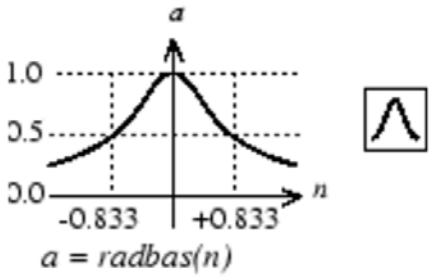
RBFN FUCTION IN MATLAB

Syntax: A = radbas(N,FP)

radbas.m function is a radial basis transfer function. It calculates a layer's output from its net input.

Inputs:

- N : S-by-Q matrix of net input (column) vectors



Radial Basis Function

```
a = radbas(n) = exp(-n^2)
```


PURDUE UNIVERSITY

STEP 1 : INITIALIZE RBFN

1. Use GenerateTrainingData.m function to generate the training inputs x, training inputs d, checking inputs Cx and checking outputs Cd.

2. Set initial parameters of RBFN

```
tdnum = 5000;          % training data number
td = x;                % training data
to = d;                % training outputs
cdnum = 1000;          % checking data number
cd = Cx;               % checking data
co = Cd;               % checking outputs
tdd = 3;               % training data dimensions
HiddenNum = 500;       % hidden nodes number
```


PURDUE UNIVERSITY

STEP 2 : PERFORM THE TRAINING

1. Determine the Centers c_i in Gaussian Function for each input using K-means algorithm.
   a. initialize centers c_i (i=1,2,3…h=number of hidden nodes) with h random selected training data
   b. calculate Euclidean distance between each input and c_i, then redistribute all the inputs to each clustering set θ_i with minimum distance.
   c. calculate the mean value of inputs in each clustering set, as new centers c_i.
   d. if $c_i(k) = c_i(k-1)$, then stop calculating new centers. $c_i(k)$ are the final centers. Otherwise, return to step b.

PURDUE UNIVERSITY

STEP 2 : PERFORM THE TRAINING

2. Calculate the Widths (Spreads) σ_i in Gaussian Function with the following equation:

$$\sigma_i = \frac{d_{\max}}{\sqrt{h}}$$

where $d_{\max}$ is maximum distance between any two centers.

3. Weights from hidden nodes to outputs can be computed by least square method, as following equation:

$$\omega = \exp\left(\frac{h}{d_{\max}}\|x_p - c_i\|^2\right)$$

PURDUE UNIVERSITY

CHOOSE 10 HIDDEN NODES FOR EXAMPLE

Choose first 10 training data of inputs as initial centers $c_i(0)$

x1	1.888	2.434	-2.238	2.480	0.794	-2.414	-1.329	0.281	2.745	2.789
x2	-1.606	1.438	2.333	2.158	0.582	0.928	2.490	-0.400	-1.261	0.791
x3	0.487	-2.576	-0.754	-2.798	-0.563	0.947	0.451	2.855	-0.350	1.667

First iteration $c_i(1)$

x1	1.070	1.883	-1.926	1.511	0.100	-2.171	-0.718	-0.270	2.174	2.080
x2	-2.057	0.856	1.464	2.543	0.101	-0.438	2.315	-0.903	-1.580	1.421
x3	0.517	-2.172	-1.738	-2.033	-1.121	0.684	1.068	2.192	-1.502	1.636

.

. after k-1 iterations

k-1 iteration $c_i(k-1)$

x1	1.774	1.374	-1.563	1.242	-1.419	-2.000	-1.605	-0.786	1.599	1.412
x2	-1.595	0.146	1.401	2.102	-1.624	-1.527	1.534	-1.290	-1.955	1.617
x3	1.486	-1.314	-1.59	-1.614	-1.769	0.586	1.431	1.980	-1.542	1.469

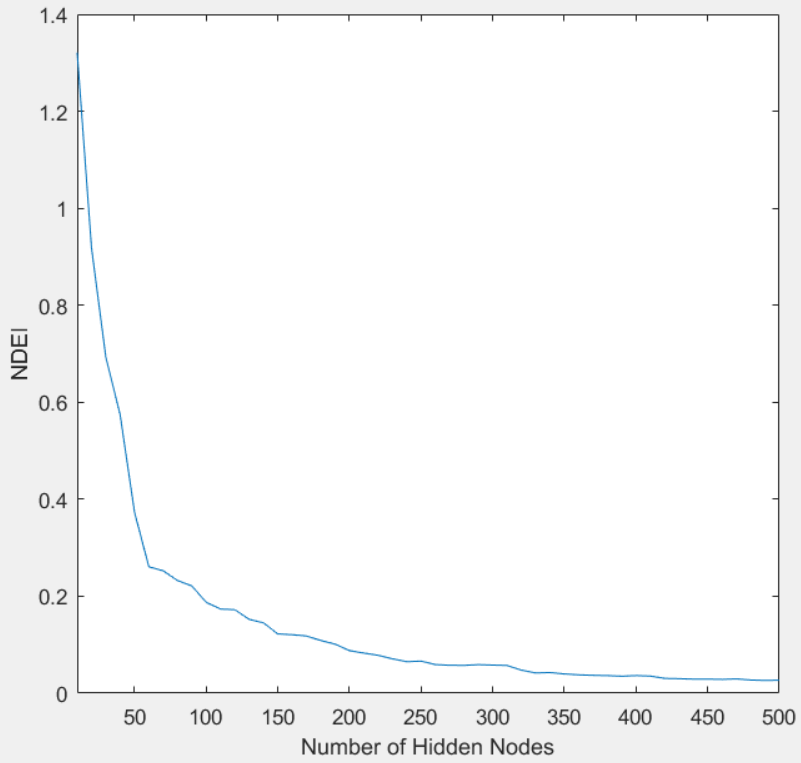
k iteration $c_i(k)$

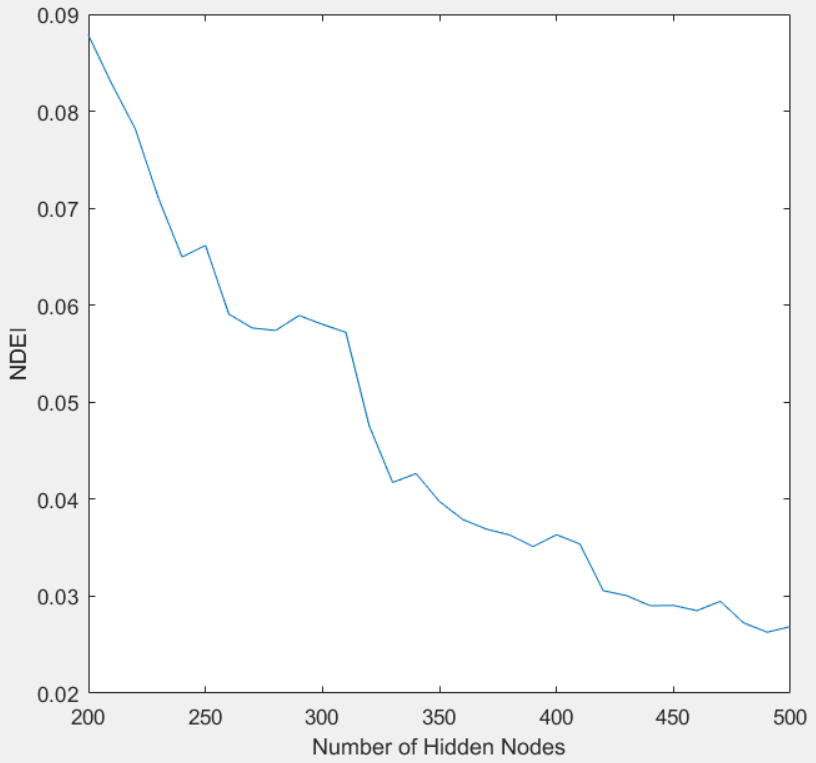
x1	1.774	1.374	-1.563	1.242	-1.419	-2.000	-1.605	-0.786	1.599	1.412
x2	-1.595	0.146	1.401	2.102	-1.624	-1.527	1.534	-1.290	-1.955	1.617
x3	1.486	-1.314	-1.59	-1.614	-1.769	0.586	1.431	1.980	-1.542	1.469

PURDUE UNIVERSITY

TESTING RESULTS

ERROR REDUCTION





No. of Hidden nodes	10	40	80	100	200	300	400	500
NDEI	1.321	0.574	0.232	0.188	0.088	0.058	0.036	0.027

PURDUE UNIVERSITY

CONCLUSION

In ANFIS, the fitting error reduced with the increasing number of membership functions. However, the more membership functions FIS has, the more time will be used to training. In this case, three inputs generated 9*9*9=729 rules in FIS with 9 membership functions and it cost more than 16 hours to tune the parameters of MFs and train the system. Since the error reduction was not significant after 6 MFs, it would be better to choose 6 MFs in this case, which spent nearly1 hour to train the system.

No. of MFs	2	3	4	5	6	7	8	9
NDEI	0.5658	0.2788	0.0339	0.0064	0.0032	0.0030	0.0042	0.0034

PURDUE UNIVERSITY

CONCLUSION

In RBFN, the fitting error reduced with the increasing number of hidden nodes. Although RBFN did not perform so well as ANFIS in this case, its error was small enough and only spent 3 minutes. Most time of training was used to compute the centers of Gaussian basis function.

No. of Hidden nodes	10	40	80	100	200	300	400	500
NDEI	1.321	0.574	0.232	0.188	0.088	0.058	0.036	0.027

To sum up, both ANFIS and RBFN can fit the nonlinear function well. In terms of training speed, RBFN is better suited for real-time control than ANFIS. However, if the number of membership functions can be selected appropriately, the speed and precision will be obtained together in ANFIS.

THANKS!

PURDUE UNIVERSITY