Fuzzy Systems - Regression

Nikolaos Giakoumoglou 9043

31/08/2020

Abstract

The aim of this assignment is to investigate the ability of TSK models to model multivariate, nonlinear functions. Specifically, two datasets are selected from the UCI repository in order to estimate the variable target from the available data, using fuzzy neural models. The first dataset will be used for a simple investigation of the training and evaluation process of such models, as well as a demonstration of ways of analyzing and interpreting the results. The second, more complex dataset will be used for a more complete modeling process, which will include, among other things, preprocessing steps such as feature selection, as well as methods for optimizing models through cross validation.

Contents

1 Application to a Simple Dataset

1.1 Preparation

For this part of the assignment, we have a dataset from UCI repository called Airfoil Self-Noise which contains 1503 instances and 6 features

- Frequency
- Angle of Attack
- Chord length
- Free steam velociry
- Suction side displacement thickness
- Scaled sound pressure level

Four TSK models are trained according to the following matrix:

	Number of MFs	Output type
TSK Model 1	2	Singleton
TSK Model 2	3	Singleton
TSK Model 3	2	Polynomial
TSK Model 4	3	Polynomial

Table 1: TSK models

All four models will be trained with the hybrid method according to which the parameters of the membership functions are optimized through the back-propagation (BP) algorithm and the parameters of the output are optimized through least squares method. All membership functions are bell-shaped with 50% overlap.
For the evalutation of the model we used 5 metrics: MSE, $RMSE$, R^2, $NMSE$ and $NDEI$. The output is concidered to be the scaled sound pressure level.
The procedure is the same for each model:

1. Clear workspace
2. Load airfoil self noise dataset
3. Preprocess the dataset (`MATLAB`: `split_scale`): shuffle and split into 3 non-overlapping sets: training, validation and check
4. Generate FIS with bell-shaped MFs (2 or 3) and specific output (constant or polynomial)
5. Tune the FIS (`MATLAB`: `anfis`)

6. Plot results: learning curve, prediction error and the 5 metrics using the the tuned FIS for which the validation error is minimum (`MATLAB` : `chkFIS`)

In the following subsections we present the initial configuration of the membership functions, the membership functions after the tuning, the learning curve, the prediction error and the 5 metrics.

1.2 TSK Model 1 (script `TSK_model_1.m`)

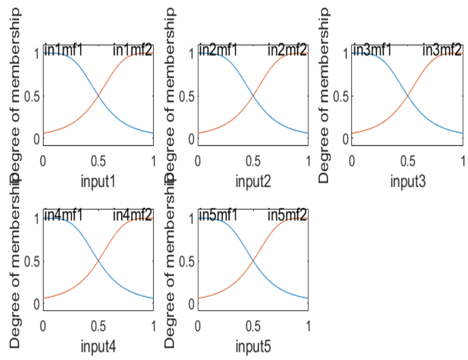

Figure 1: Initial MFs

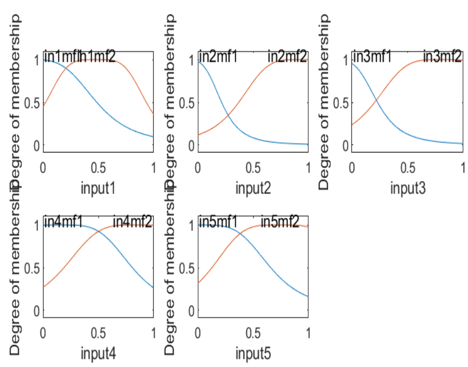

Figure 2: MFs after tuning

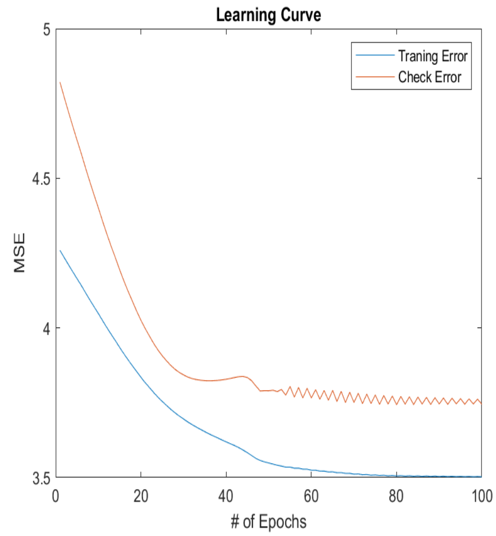

Figure 3: Learning curve (overfitting at epoch ~40)

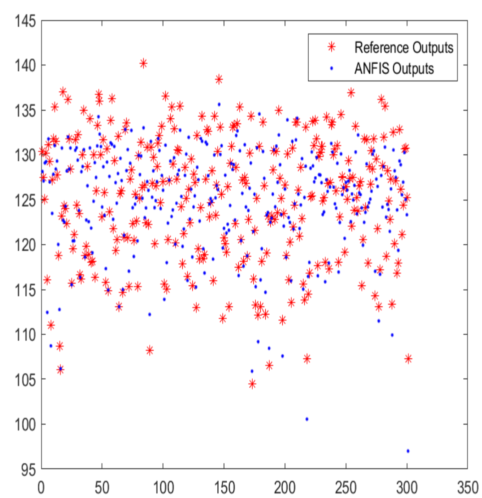

Figure 4: Reference output VS ANFIS output

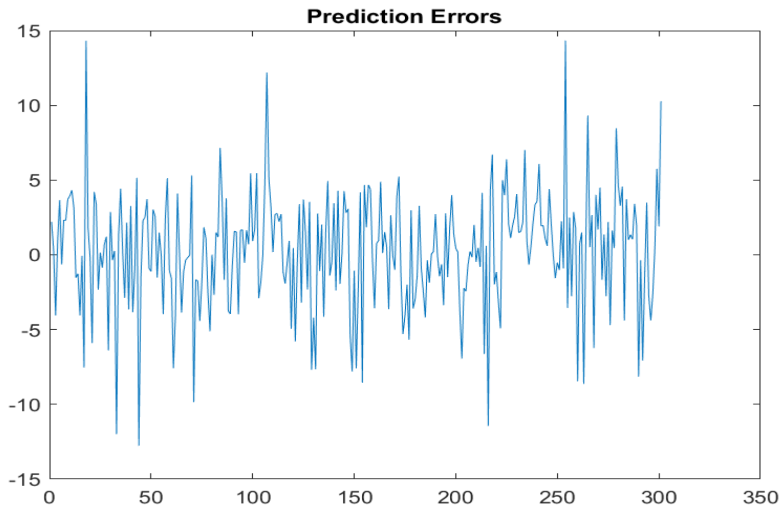

Figure 5: Prediction errors

MSE	15.078
$RMSE$	3.883
R^2	0.689
$NMSE$	0.309
$NDEI$	0.556

Table 2: Metrics

1.3 TSK Model 2 (script `TSK_model_2.m`)

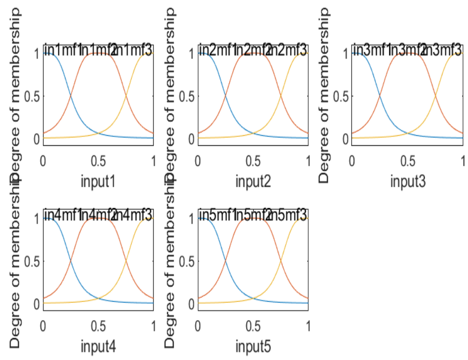

Figure 6: Initial MFs

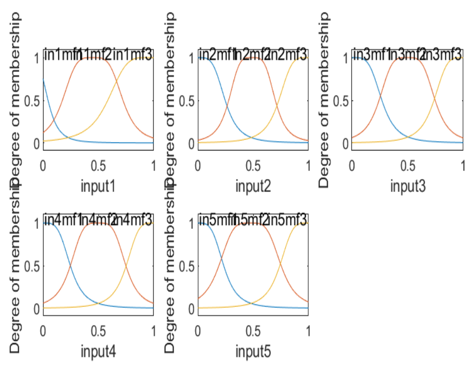

Figure 7: MFs after tuning

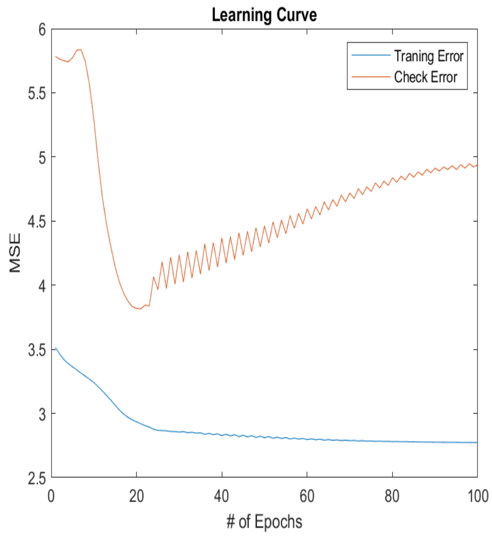

Figure 8: Learning curve (overfitting at epoch ~20)

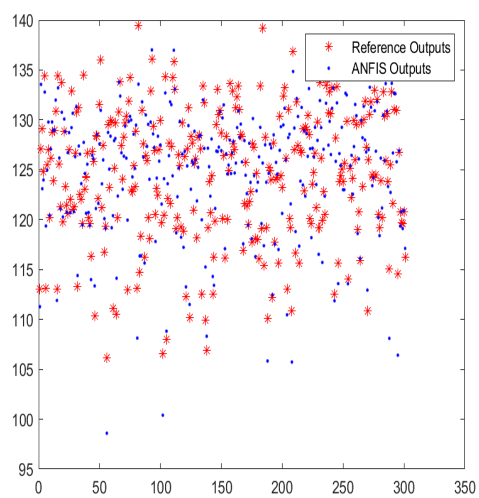

Figure 9: Reference output VS ANFIS output

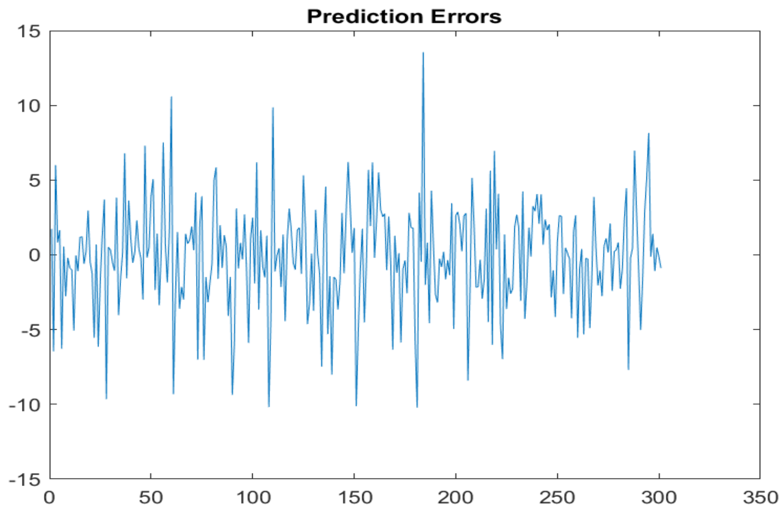

Figure 10: Prediction errors

MSE	12.865
$RMSE$	3.586
R^2	0.708
$NMSE$	0.290
$NDEI$	0.539

Table 3: Metrics

1.4 TSK Model 3 (script TSK_model_3.m)

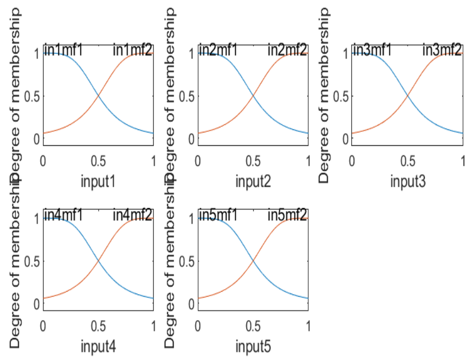

Figure 11: Initial MFs

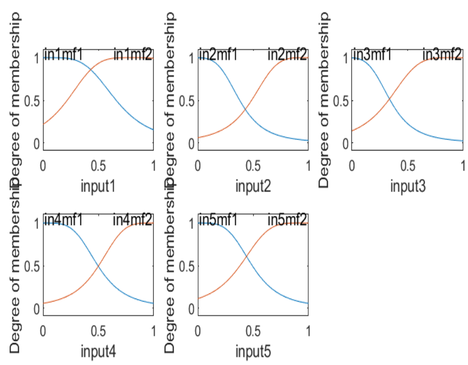

Figure 12: MFs after tuning

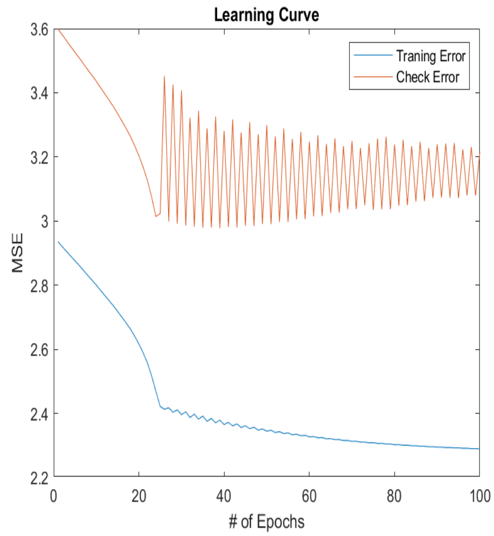

Figure 13: Learning curve (overfitting at epoch ~25)

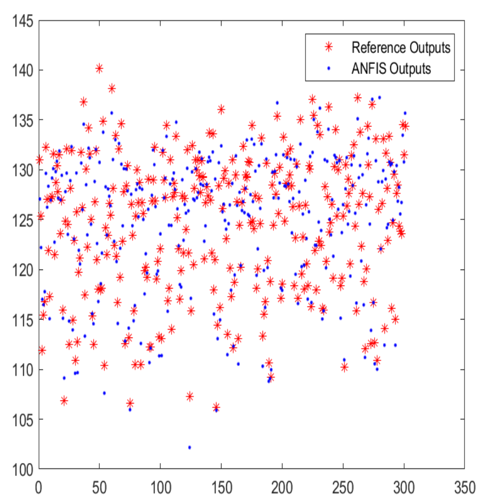

Figure 14: Reference output VS ANFIS output

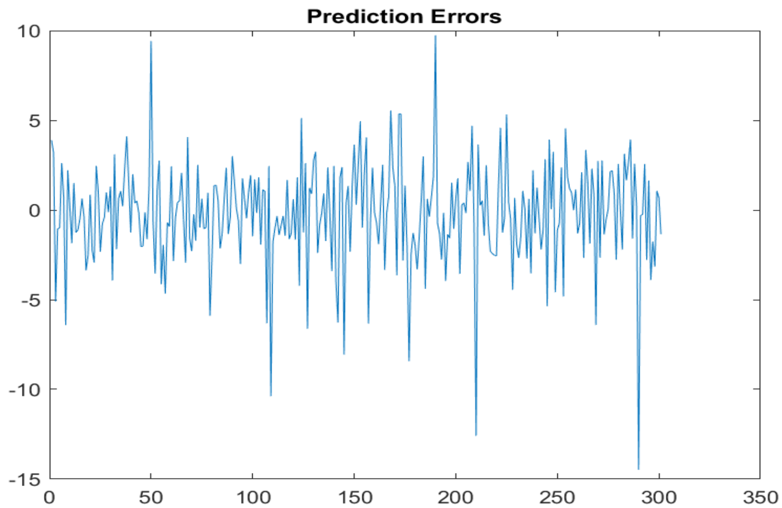

Figure 15: Prediction errors

MSE	9.4161
$RMSE$	2.877
R^2	0.834
$NMSE$	0.165
$NDEI$	0.406

Table 4: Metrics

1.5 TSK Model 4 (script TSK_model_4.m)

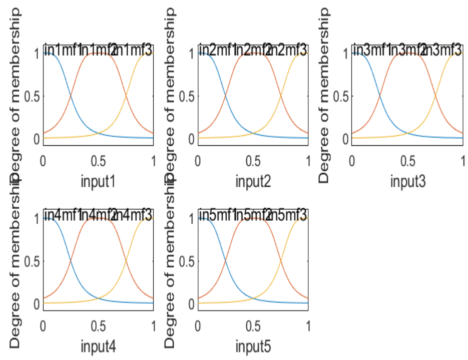

Figure 16: Initial MFs

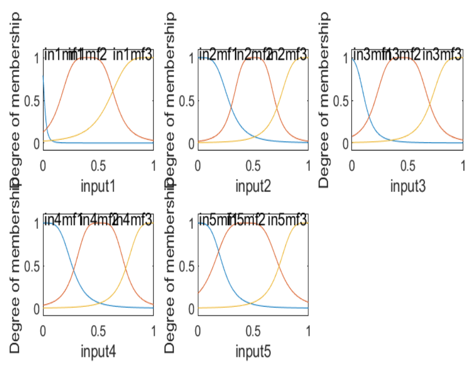

Figure 17: MFs after tuning

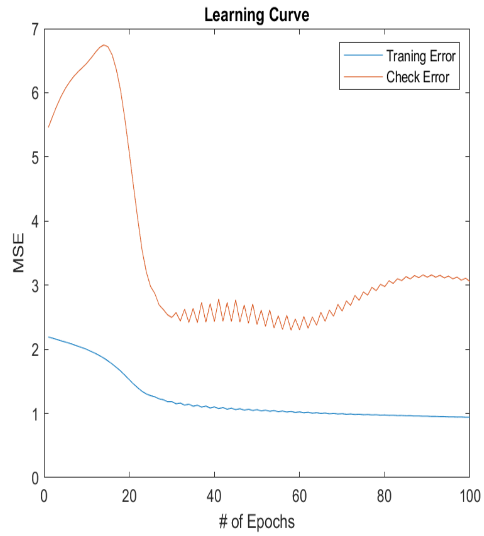

Figure 18: Learning curve (overfitting at epoch ~20)

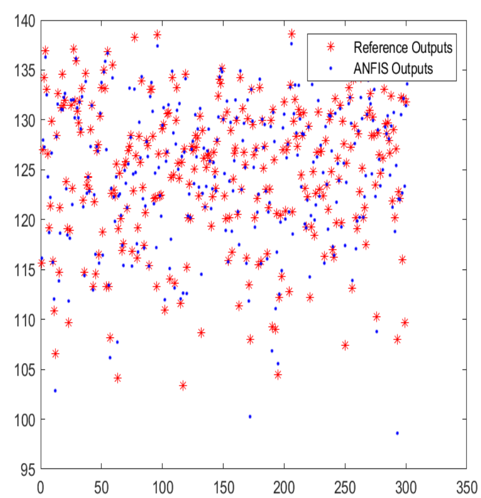

Figure 19: Reference output VS ANFIS output

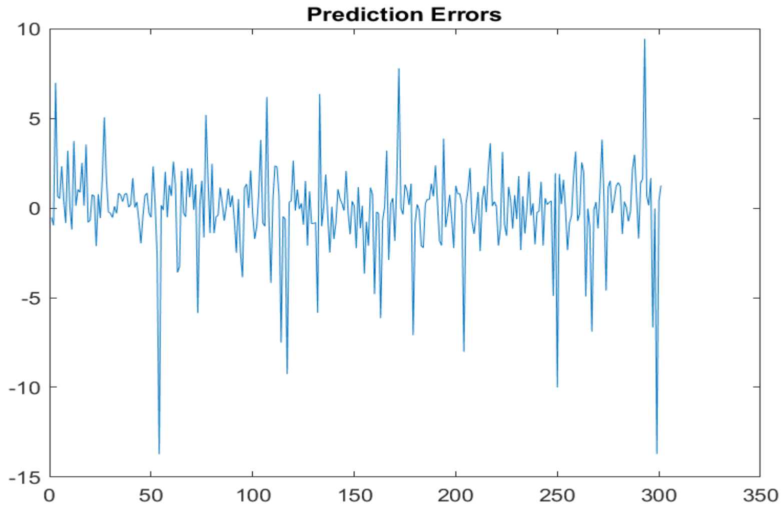

Figure 20: Prediction errors

MSE	6.442
$RMSE$	2.538
R^2	0.877
$NMSE$	0.122
$NDEI$	0.349

Table 5: Metrics

1.6 Conclusion

All 4 models with their metrics are concentrated in the following table:

	No MFs	Output type	MSE	$RMSE$	R^2	$NMSE$	$NDEI$
TSK Model 1	2	Singleton	15.07	3.88	0.68	0.30	0.55
TSK Model 2	3	Singleton	12.86	3.58	0.70	0.29	0.53
TSK Model 3	2	Polynomial	8.27	2.87	0.83	0.16	0.40
TSK Model 4	3	Polynomial	6.44	2.53	0.87	0.12	0.34

Table 6: All 4 models and their metrics

TSK Models 3 and 4 are much better than 1and 2. The MSE is minimized for the TSK Model 4, and the R^2 is the best (closest to 1), so we can say that this model is the best among the four. Moreover, the MSE and RMSE are the lowest among the 4 models which indicate lower error, so as the NDE and NMSE which indicate a better fit. However TSK Model 4 took a long time to train ($\sim$ 1h) and the improvement from Model 3 is not that significant (see MSE and R^2), which took less than 5 minutes to be trained. We also observe that the more MFs we use, the best the model is and the polynomial output type gives better results in general. On the contraty, models with singleton output take some minutes to be trained.

2 Application to a dataset with high dimensionality

2.1 Preparation

For this part of the assignment, we have a dataset from UCI repository called Superconductivity which contains 21263 instances and 81 features. The features are related to superconductive materials and the output is the critical temperature. This is a high-dimension dataset which makes the application of a TSK model prohibitive; with just 2 MFs, the number of rules would be 2^{81}! We will use a different approach with the grid search algorithm.
The procedure is the following

1. Clear workspace
2. Load superconductivity dataset
3. Preprocess the dataset (`MATLAB` : `split_scale`): shuffle and split into 3 non-overlapping sets: training, validation and check
4. Feature selection with 100 nearest neighbours using Relief algorithm (`MATLAB`: `relieff`) for the shuffled data
5. Arbitrarily define values for the optimal number of features and radius of cluster center's range of influence in each of the data dimensions, assuming the data falls within a unit hyperbox
6. For each pair of feature and radius (the two variables form a grid which we search for the optimal model) we generate the FIS (`MATLAB`: `genfis2`), perform 5-Fold Cross Validation (`MATLAB` : `cvpartition`: by default training set is 80% and check set is 20% as given by the indexes) by tuning the FIS (`MATLAB` : `anfis`) for each fold. Each fold represents a secondary model which we use to find an error using the validation set. The mean of all 5 errors of the folds represent the error of the initial model (specific value for rules and features). This procedure is called Grid Search
7. Plot the errors of the models in 2D and 3D
8. Find the optimal model; the one with the minimum MSE

2.2 Find Optimal Model (script `grid_search.m`)

In this case we use $3, 9, 15, 21$ features with radius from the set $[0.15, 0.25, 0.35, 0.45, 0.55, 0.65, 0.75, 0.85, 0.95]$.
In order to make thinks parallel, we split the initial scirpt and run it in 4 different instances of `MATLAB` (see the folder *grid_search_bf*). The duration of the scripts were 2 hours.
Each cell is based on the following table

Features	Radius								
3 Features	0.15	0.25	0.35	0.45	0.55	065	0.75	0.85	0.95
9 Features	0.15	0.25	0.35	0.45	0.55	065	0.75	0.85	0.95
15 Features	0.15	0.25	0.35	0.45	0.55	065	0.75	0.85	0.95
21 Features	0.15	0.25	0.35	0.45	0.55	065	0.75	0.85	0.95

Table 7: Values of features and radiuses used

We will now present the results of the MSE in 3 different forms (table, 2D and 3D figures).

Features	Radius Set								
3 Features	357	387	396	443	464	482	–	–	–
9 Features	263	234	253	265	284	299	302	306	316
15 Features	269	244	222	233	236	236	261	263	293
21 Features	–	196	181	207	219	215	231	221	250

Table 8: MSE in table form

The $-$ indicate that either the number of rules was too low or too high.

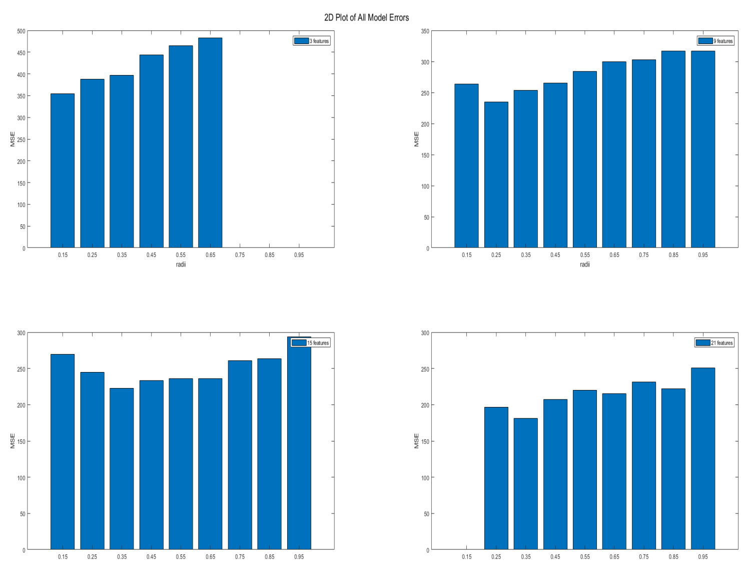

Figure 21: MSE in 2D

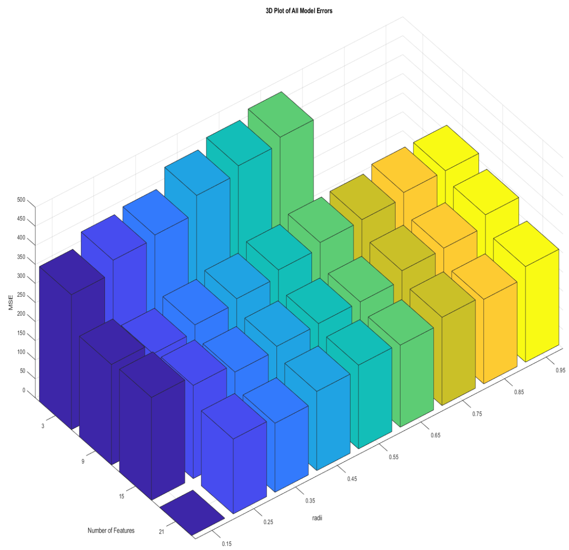

Figure 22: MSE in 3D

The minimum MSE is 181.1869 for 21 features and radius 0.35, which we save in `opt_model.mat` We observe that while we increase the number of features, the MSE decreases. Also, if we use a small radius, the MSE tends to be lower (except from 15 features). For the best visualiazation, the MSE in 3D is quite helpful.

2.3 Optimal Model (script `opt_model.m`)

Based on `grid_search.m` the optimal model appears to be the one with

- MSE equals to 181.1869
- Number of features: 21
- Radius: 0.35
- Number of rules: 9

We load the model which we already saved in `opt_model.mat` and train the ANFIS. Results are the following

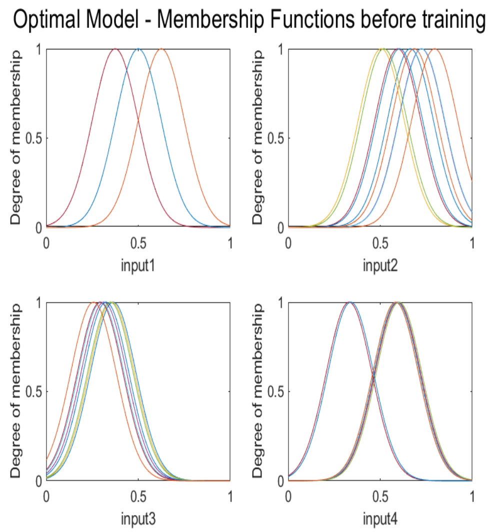

Figure 23: MFs before training

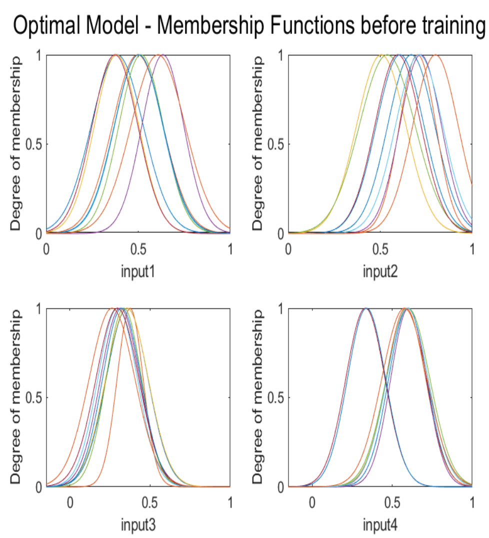

Figure 24: MFs after tuning

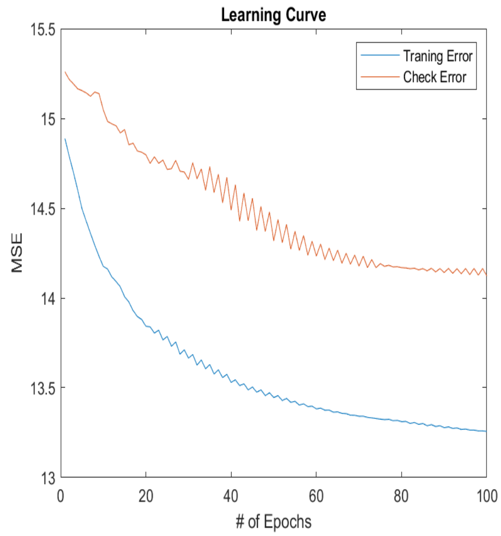

Figure 25: Learning curve

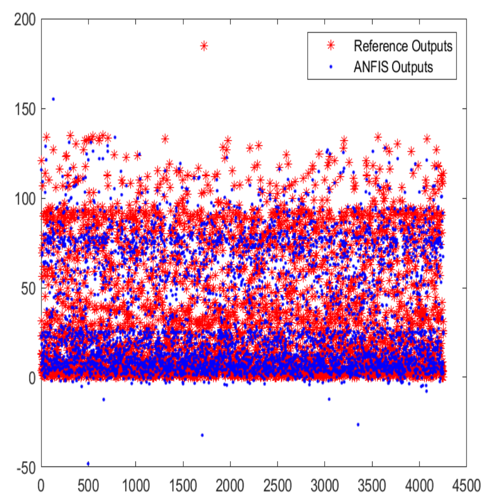

Figure 26: Reference output VS ANFIS output

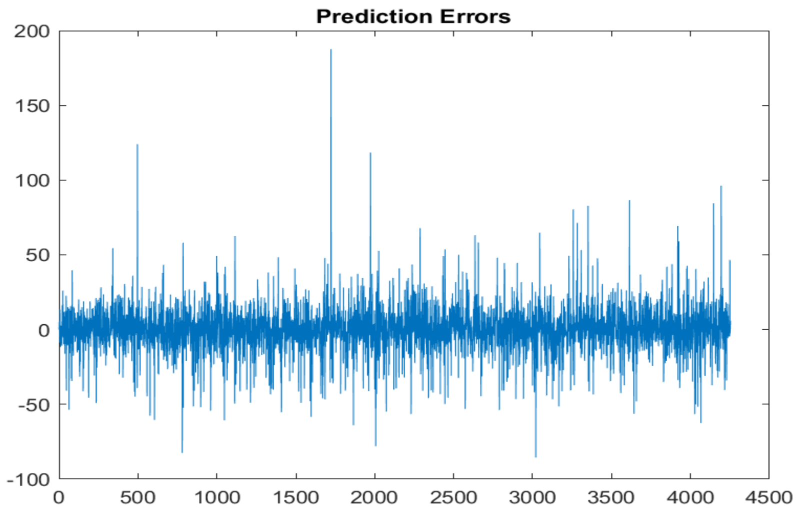

Figure 27: Prediction errors

MSE	221.17
$RMSE$	14.875
R^2	0.81438
$NMSE$	0.18558
$NDEI$	0.43079

Table 9: Metrics

2.4 Conclusion

The number of rules is significant less than the 2^{81} (for 2 membership function) as we mentioned before. Even if we chose 15 features or 30 with grid partiotioning, the number of rules would be $2^{15}$and 2^{30}respectively. For 3 membership function, that number would be $3^{15}$and 3^{30}! Now we only have 9 rules!
The accuraccy of the models is quite good based on the R^2 which is around 0.8 (ideal R^2 is one). The membership functions after training do not seem to change much, rather than move to the right. The values of the number of features, cluster's radius and rules that were tested were chosen arbitrarily and close to each other so we can have a global view of the results, but it would make no big difference for other values since the sets we chose were representing and the scripts were already computational demanding.

List of Figures

List of Tables