Predicting Diabetes using Adaptive-Network-based Fuzzy Inference System (ANFIS)

Krishna Chaitanya Reddy Tamataam
Department of Mathematics
IIT Delhi
Delhi 110016, India
mt6180785@maths.iitd.ac.in

Subhalingam D
Department of Mathematics
IIT Delhi
Delhi 110016, India
mt1180770@maths.iitd.ac.in

Abstract

We studied the Adaptive-Network-based Fuzzy Inference System (ANFIS) proposed by Jang (1993). We implemented a simple version of ANFIS for Diabetes predictor with Gaussian membership functions and tuned it. Our model had a test accuracy of 81.30%, train accuracy of 85.42%, test F1-score of 74.72% and train F1-score of 78.44%. We further analysed the variation of performance metrics and losses with epochs, constructed the confusion matrix, plotted graphs of the membership functions learnt by the model and discussed our observations.

1 Introduction

In the recent times, due to availability of large amount of data and high computational power, Artificial Intelligence (AI) and Machine Learning (ML) algorithms play a very important role in automation. From basic neural networks to predict a simple non-linear functions to speech recognition using Recurrent Neural Network (RNN), there are several applications of AI that we use in our day-to-day lives. In today's world these models have been trained rigorously and have achieved high accuracy. But a feature used for classification cannot be a crisp value exactly, i.e., there is some amount of *fuzziness* involved (Zadeh, 1996). To account for this, we use membership function to estimate the relatedness of an element to a set. This feature plays a very important role in identifying unclear examples in the dataset. Fuzzy neural networks (FNN) combine the Fuzzy Logic and Neural networks in single model. We study Adaptive-Network-based Fuzzy Inference System (ANFIS) (Jang, 1993) and implement a model to that predicts whether a person is having diabetes or not. ANFIS has many applications in field of geo-engineering studies, such as soil analysis (predicting friction angles of soil), variations in salinity of sea with temperature, pressure and other factors. In the field of image detection, it is used for identifying Breast cancer.

2 Model

2.1 Architecture

Adaptive-Network-based Fuzzy Inference System (ANFIS) is derived from the first order Takagi-Sugeon fuzzy inference system (Lohani et al., 2006). The main advantage of such a system over other inference system is that the parameters used for its membership functions can be changed (different membership functions, described in Section 2.3, can be used for each fuzzy rule). Hence we can apply Machine Learning (ML) algorithms to train these parameters and build a model that fits into the given dataset. Moreover, ANFIS can be stacked with any other Deep Learning models, which makes it more interesting to study and unique from other Fuzzy Neural Networks.

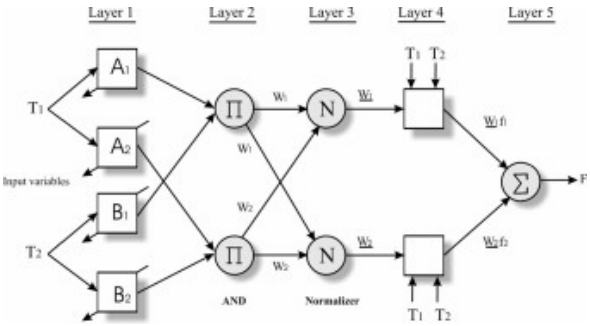


Figure 1: Architecture of ANFIS

ANFIS is made up of *five* layers as discussed in the following subsections.

2.1.1 Input layer

This contains the input features of the data used by the model. The data has to be standardised before it is being fed to the model. Mathematically, our input data will be a n dimensional vector for each example (data point).

2.1.2 Fuzzification Layer

This layer is made up of mn nodes (where m is the desired number of fuzzy rules and n is the number of input features). For every input feature i, $(i = 1, 2, \ldots, n)$ there are nodes a_{ij} where $j = 1, 2, \ldots, m$. A Gaussian membership function ([13] in Section 2.3), with trainable parameters μ (mean) and σ^2 (variance), is applied to the input features to obtain the output for this layer.

2.1.3 Rule Layer

. This layer takes the product (algebraic intersection, $\cap_a$) of the respective rules obtained in the previous layer. The output of this layer is fed as input for the next layer.

Each node a_j in this layer, it2 takes $x_{1j}, x_{2j}.....x_{nj}$ from the previous layer, where n is the number of features and $j = 1, 2, \ldots, m$, where m is the number of fuzzy rules. It gives product of all these values, i.e., $a_j = \prod_{i=1}^{n} x_{ij}$, as the output which is then normalised.

2.1.4 De-fuzzification layer

It has m nodes which is the number of fuzzy rules. As the name says, this layer is used for defuzzification. Each node takes the corresponding previous node output and the input layer values and does the following $(a_0 + a_1x_1 + a_2.x_2 + \cdots + a_nx_n).w$ where w is the weight of the corresponding rule layer and $a_0, a_1, a_2, \ldots, a_n$ are trainable parameters.

2.1.5 Output layer

All the outputs from previous layer is added and an activation function (sigmoid function, in our case) is applied.

2.2 Equations for Forward pass

2.2.1 Layer 1

For n input features x_i $(i = 1, 2, \ldots, n)$,

$$\text{Input: } x_1, x_2, \ldots, x_n \tag{1}$$

$$\text{Output: } x_1, x_2, \ldots, x_n \tag{2}$$

2.2.2 Layer 2

For each node a_{ij} where $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$

$$\text{Input: } x_i \tag{3}$$

$$\text{Output: } N_{i1}(x_i), N_{i2}(x_i), ...N_{im}(x_i) \tag{4}$$

where N_{ij} is the Gaussian membership function ([13] in Section 2.3) with mean μ_{ij} and variance σ_{ij}^2.

2.2.3 Layer 3

For each node j $(j = 1, 2, \ldots, m)$,

$$\text{Input: } w_{1j}, w_{2j}, \ldots, w_{nj} \tag{5}$$

$$w_j = w_{1j} \cdot w_{2j} \cdots w_{nj} \tag{6}$$

$$sum = w_1 + w_2 + \cdots + w_m \tag{7}$$

$$\text{Output: } w_j^* = w_j / sum \tag{8}$$

Here, w_j^* is the normalized weight.

2.2.4 Layer 4

for each j $(j = 1, 2, \ldots, m)$,

$$\text{Input: } f_j = a_{0j} + a_{1j}x_1 + \cdots + a_{nj}x_n \tag{9}$$

$$\text{Output: } w_j^* \dot{f}_j \tag{10}$$

where $a_0, a_1, \ldots, a_n$ are trainable parameters.

2.2.5 Layer 5

for each j $(j = 1, 2, \ldots, m)$

$$\text{Input: } X = \sum_{j=1}^{m} w_j^* \dot{f}_j \tag{11}$$

$$\text{Output: } \sigma(X) \tag{12}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function. If $x > 0$, $\sigma(x) > 0.5$, so we assign a label 1 to such an input (and 0 otherwise).

2.3 Membership functions

Gaussian membership function is the widely used membership function. Yet, other membership functions like Generalized Bell membership function, Triangular fuzzy number, Trapezoidal fuzzy number, etc., can also be applied. However, it is recommended to choose a function with smooth curves, so that they are differentiable (helpful for optimizer while training), and it also helps us to learn nonlinear functions. In Figure 2, 2a represents Gaussian membership function, 2b represents a combination of two Gaussian functions and 2c represents Generalised bell function.

Given below is Gaussian Membership Function we used in the Fuzzification Layer (Section 2.1.2)

$$N(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{13}$$

where μ is the mean and σ^2 is the variance.

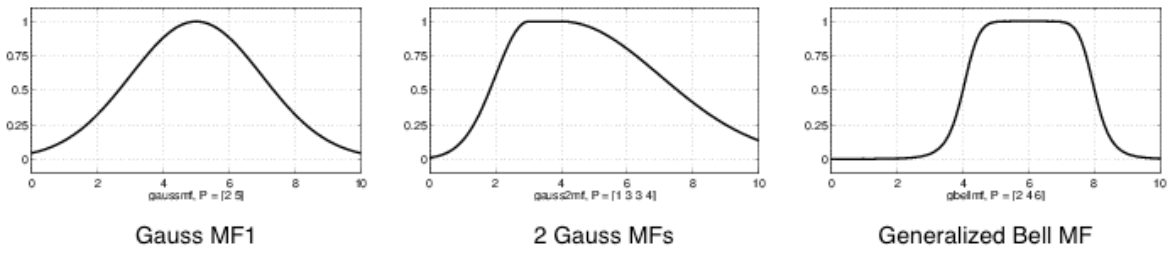


Figure 2: Different Membership Functions that can be used

3 Experiment

3.1 Dataset

We use the "*Pima Indians Diabetes Database*" from Kaggle [1]. The dataset contains 768 examples and 8 features (*plus one* outcome) as mentioned in Table 1. *It is worthy to note that this dataset only consists of females who are at least 21 years old.*

Feature	Short Description
Pregnancies	Number of times pregnant
Glucose	Plasma glucose concentration a 2 hours in an oral glucose tolerance test
Blood Pressure	Diastolic blood pressure (mm Hg)
Skin Thickness	Triceps skin fold thickness (mm)
Insulin	2-hour serum insulin (mu U/ml)
BMI	Body mass index (weight in kg/(height in m)^2)
Diabetes Pedigree Function	Diabetes pedigree function
Age	Age (years)
Outcome	Class variable (0 or 1)

Table 1: Description of features in Dataset

3.2 Pre-processing

We split the dataset into train and test data using `sklearn.model_selection.train_test_split()` The model requires each feature to have a mean 0 and standard deviation 1. Hence, we standardize each column of the dataset before using it.

It was observed that the dataset did not have any missing data-so no additional processing were necessary.

3.3 Settings

We conducted our experiment with the following hyperparameters:

- The model was implemented[2] in Python using TensorFlow (v1.15.4) (Abadi et al., 2015). Scikit-learn (v0.24.0) (Pedregosa et al., 2011) was used to compute the metric values and Matplotlib (v3.0.3) (Hunter, 2007) was used to plot graphs.
- As mentioned in Section 3.2, we used `sklearn.model_selection.train_test_split()` with `test_split` set to 0.16. Moreover, `random_state` was set to 2021.
- Total number of fuzzy rules used in the model was set to 14.
- Binary Cross Entropy was the loss function used.
- Adam Optimizer was used with learning parameter $\alpha = 0.01$ to optimize the loss function in 1000 epochs.
- Data points for plotting graphs in Section 3.4 were taken at every 10 epochs.
- Gaussian membership function was used for each fuzzy rule. The weights, μ, σ were randomly initialised from normal distribution.

Additionally, we omit or discard the *Pregnancies* column from the dataset, while building our model, as we thought it doesn't have a direct relation with the disease of interest.

[1] https://www.kaggle.com/uciml/pima-indians-diabetes-database

[2] Code available at https://github.com/subhalingamd/ANFIS

After splitting the dataset, we analysed the number of data points belonging to each class in train and test set and mentioned it in Table 2.

	#1 (Positive)	#0 (Negative)
Train	225	420
Test	43	80
Total	268	500

Table 2: Distribution of data in train and test set

3.4 Results

We use the following notations to define the performance metrics we use in this report:

- **TP** (True Positive): Model predicts Positive and it is actually Positive.
- **FP** (False Positive): Model predicts Positive but it is actually Negative.
- **FN** (False Negative): Model predicts Negative but it is actually Positive.
- **TN** (True Negative): Model predicts Negative and it is actually Negative.

Now, we define the performance metrics of our interest.

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN}$$

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Recall} = \frac{TP}{TP+FN}$$

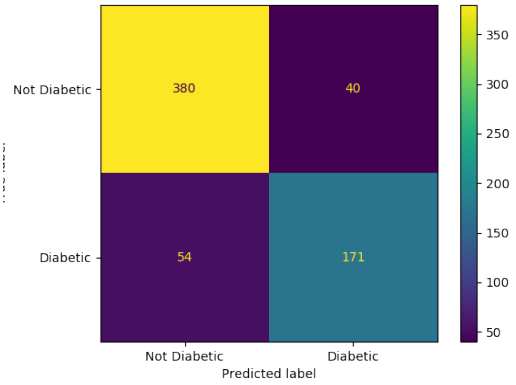
$$\text{F1-Score} = \frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$$

The performance and losses of model on train and test set is given in Table 3.

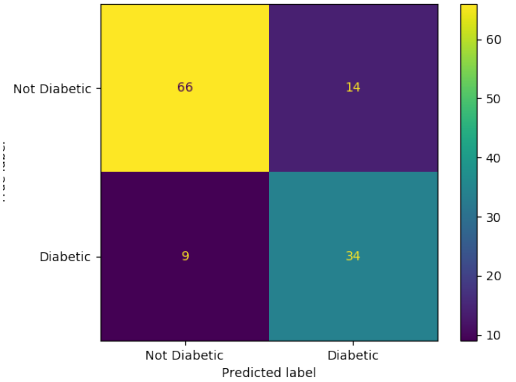
	Test	Train
Accuracy	81.30%	85.42%
Precision	70.83%	81.04%
Recall	79.06%	76.00%
F1-score	74.72%	78.44%
Loss	0.43418	0.33549

Table 3: Performance and Losses of the model

We have constructed the confusion matrix in Figure 3 (for test set in Figure 3a and train set in Figure 3b). We plotted the losses in Figure 4 and performance metrics (accuracy, F1-score, precision, recall) in Figure 5 over number of epochs, for both test and train sets. The graph of membership functions learnt by each fuzzy rule for the given features are plotted in Figure 6.



(a) Train set



(b) Test set

Figure 3: Confusion Matrix

Additionally, the following were observed:

- The model's performance was found to be too sensitive to the initialisation of the parameters. Since we initialised them randomly, we observed that there was a lot of fluctuation in terms of accuracy and F1-score when we ran them multiple times.
- Just like any Machine Learning models, we observed that lower number of fuzzy rules led to underfitting while increasing them too much led to overfitting. This can be reasoned similar to the underfitting (or overfitting) case when we choose too less (or too many) hidden units in a Neural Network.

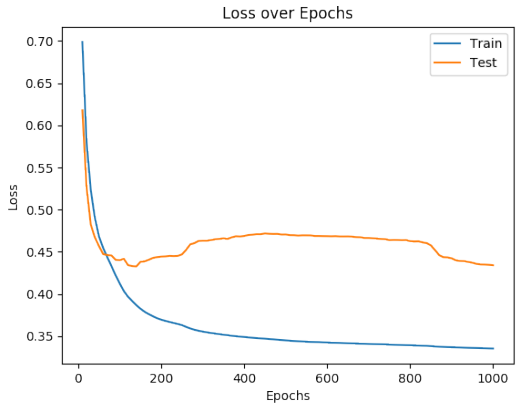


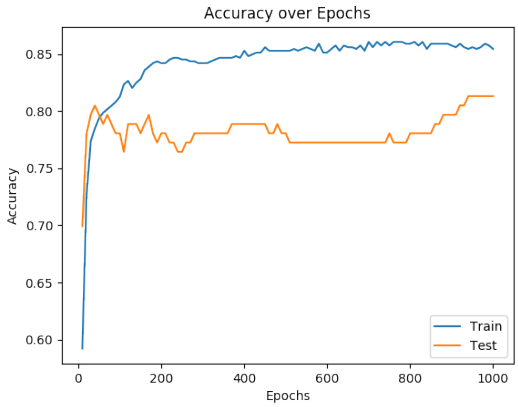
Figure 4: Loss over epochs

4 Possible Additions

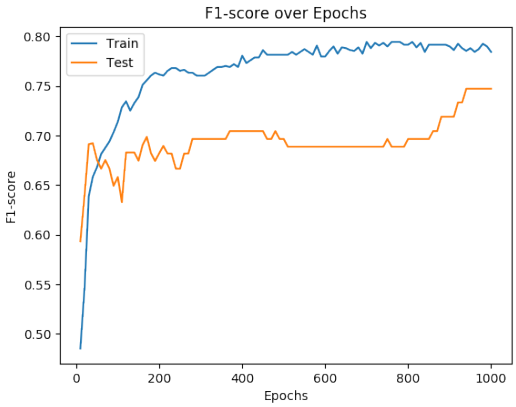
We can build a similar image classifier for breast cancer detection. However, for images, the number of input features (total number of pixels) in image will be too high. One approach would be to reduce the these dimensions and use the reduced features as the input to our model. This can be achieved with some Dimension Reduction techniques like Principal Component Analysis (PCA) (Espirito Santo, 2012).

5 Conclusion

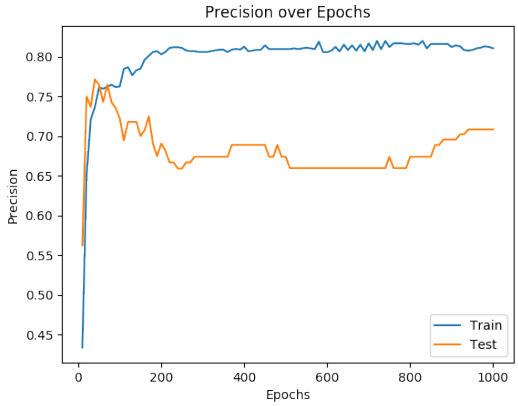
We studied the working of ANFIS and applied the concepts to a real-life dataset and reported our results in Section 3.4, after training over 100 different models. In medical domain, we expect the recall to be high, as we do not want to miss a positive case, even at the cost of false positive.



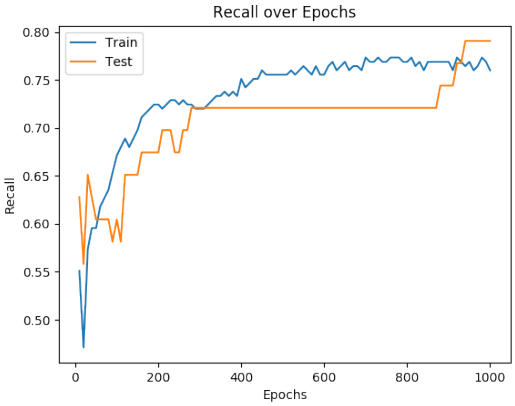
(a) Accuracy



(b) F1-score

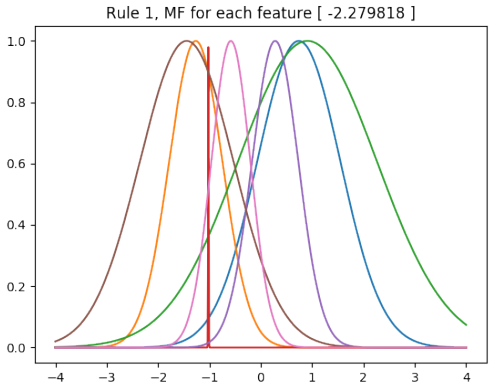


(c) Precision

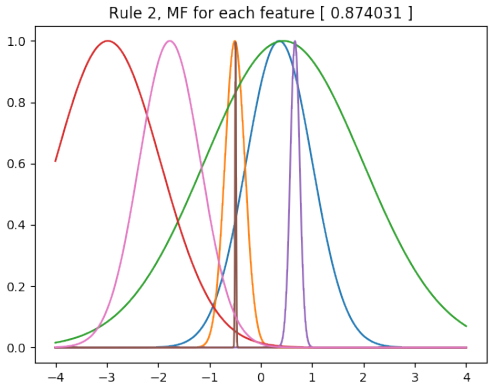


(d) Recall

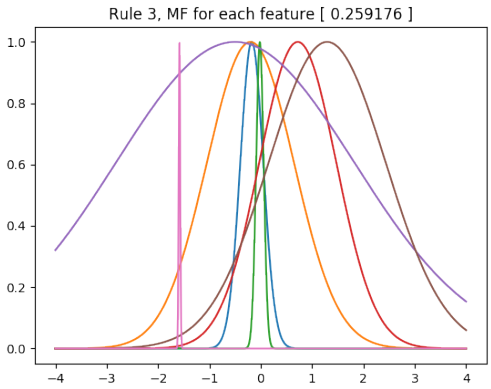
Figure 5: Performance metrics over epochs



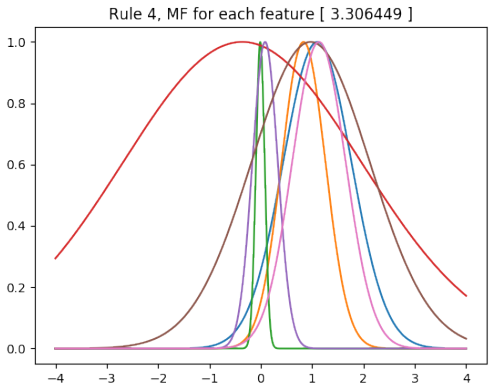
(a) Rule 1



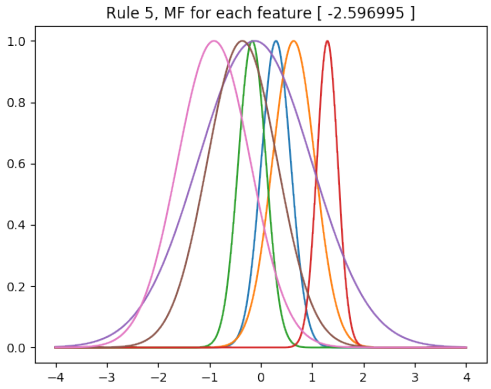
(b) Rule 2



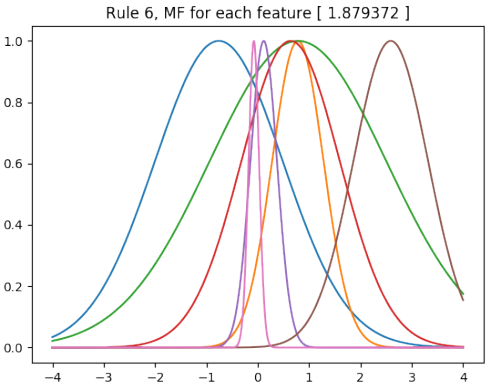
(c) Rule 3



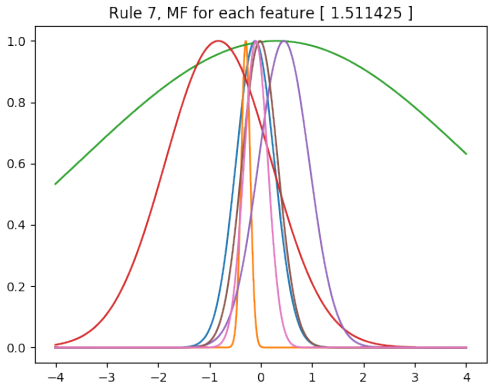
(d) Rule 4



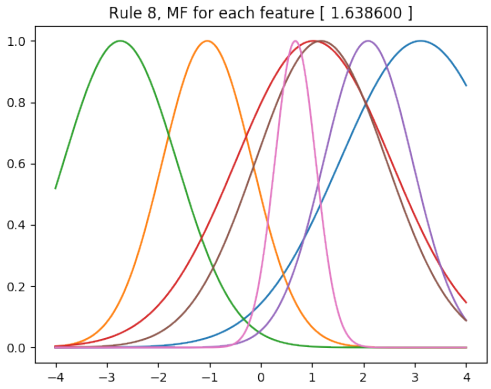
(e) Rule 5



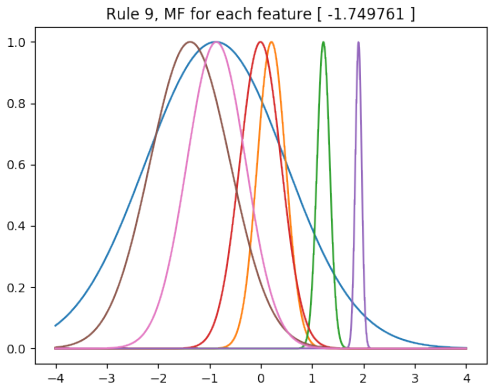
(f) Rule 6



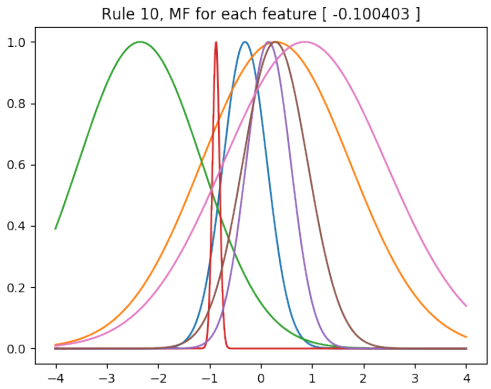
(g) Rule 7



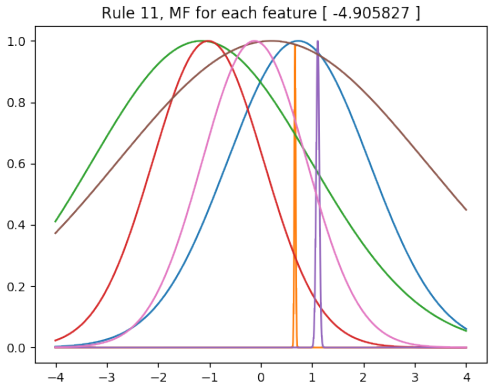
(h) Rule 8



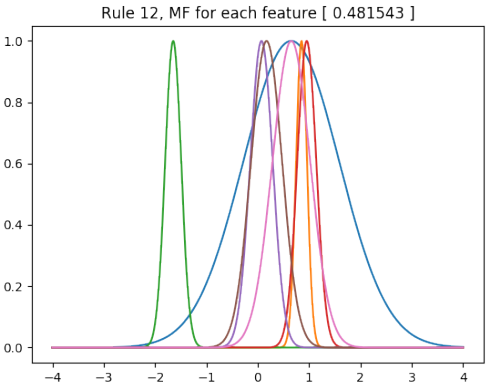
(i) Rule 9



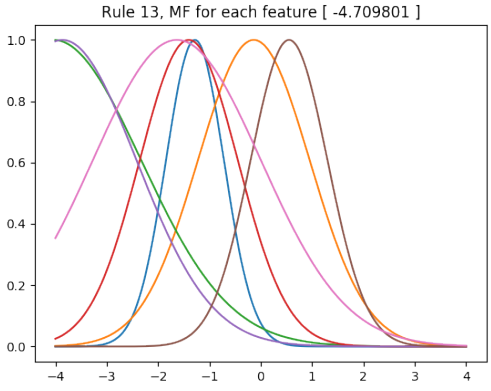
(j) Rule 10



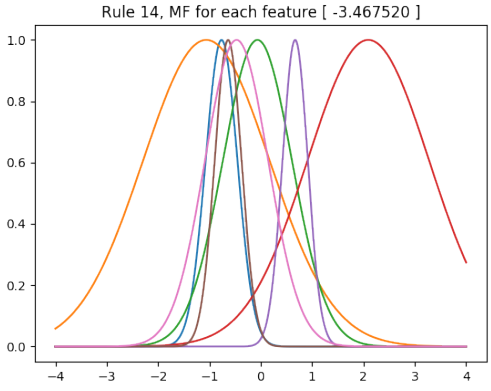
(k) Rule 11



(l) Rule 12



(m) Rule 13



(n) Rule 14

Figure 6: Membership Functions of Fuzzy Rules learnt

Acknowledgments

This project was inspired from a paper title "*Application of Adaptive Neuro-Fuzzy Inference System for diabetes classification and prediction*" by Oana Geman, Iuliana Chiuchisan, and Roxana Toderean. However, the work done in this report is ours (unless cited) and the paper mentioned above helped us to understand some applications of ANFIS and choose a dataset.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Rafael Espirito Santo. 2012. Principal component analysis applied to digital image compression. *Einstein (São Paulo, Brazil)*, 10:135–9.

J. D. Hunter. 2007. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

J. . R. Jang. 1993. Anfis: adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(3):665–685.

A.K. Lohani, N.K. Goel, and K.K.S. Bhatia. 2006. Takagi–sugeno fuzzy inference system for modeling stage–discharge relationship. *Journal of Hydrology*, 331(1):146 – 160. Water Resources in Regional Development: The Okavango River.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Lotfi A Zadeh. 1996. Fuzzy sets. In *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh*, pages 394–432. World Scientific.