Electronic Supplementary Material (ESI) for Dalton Transactions.

Electronic supplementary information

Human brain inspired chemical artificial intelligence tools for analysis and prediction of anion sensing characteristics of imidazole-based luminescent Os(II)-bipyridine complex

Sohini Bhattacharya[a], Anik Sahoo[a] and Sujoy Baitalik[a*]

Artificial neural networks (ANNs)

An artificial neural network is a network stimulated by the central nervous system of the animals, primarily the brain. ANNs are often employed to guess functions which could rely on huge number of unknown inputs. Among the two principal categories of neural networks, viz. recurrent (RNN) and feed-forward (FFN), we employed FNN in the present study due to static nature of our system. FNN is the simplest and convenient category of network where the information passes into a particular direction, proceeds, from the input nodes, via the hidden notes, and finally to the output nodes. Additionally, due to its high efficiency in forecasting static system, we implemented advanced feed-forward back propagation network, namely, ANN-function fitting (ANN-FF) network for deeper understanding and forecasting of the system.

Artificial neural network model consisting of 2 inputs, 5 hidden layers and 2 output. In ANN-FF, the relation between the input and output is assumed to be a function, which is approximated using the experimental data. The network diagram of the ANN-FF for the system can be found in Fig. S5†. It can fit multidimensional mapping problems arbitrarily well when consistent data and enough neurons are designed in the hidden layer. For function fitting of the problem, a neural network is needed to map between a data set of numeric inputs and a set of numeric targets. Hence, each pattern is assigned a number (e.g., 1, 2, 3, 4, etc.).

Different training algorithms have been used for prediction of photophysical and electrochemical responses of the complex in presence of acid and base. In this work, feedforward back propagation neural network is utilized for comparison of three different training algorithms, i.e., Levenberg–Marquardt Algorithm (LM), Scaled Conjugate Gradient (SCG) and Bayesian Regularization (BR), as regards of their capability to predict the photophysical and electrochemical data. The advantages and disadvantages of these three curve fitting training algorithms are described below.

Levenberg–Marquardt algorithm

This algorithm typically requires more memory but less time. Training automatically stops when generalization stops improving, as indicated by an increase in the mean square error of the validation samples. This method has been applied by researchers to challenging nonlinear least-squares problems in several different domains.

Scaled conjugate gradient

This algorithm requires less memory. Training automatically stops when generalization stops improving, as indicated by an increase in the mean square error of the validation samples.

Bayesian regularization

This algorithm typically requires more time, but can result in good generalization for difficult, small or noisy datasets. Training stops according to adaptive weight minimization (regularization).

In this study, a neural network for function fitting was coded in MATLAB 2018. The input data present the network, while the target data define the desired network output. Table S2† represents the current intensity outputs upon the action of 40 different combinations of two inputs (input 1= H^+ and input 2= OAc^-). Thus, the 40×2 matrix represents the static input data of 40 samples involving 2 inputs, while 40×5 matrix represents the static output data of five elements. Now, the 40 samples are divided into 3 sets of data. For Levenberg–Marquardt and Scaled Conjugate Gradient algorithm assisted training process, 70% of the data are conferred for the training and the network is corrected according to its error. Now the learning algorithm and the number of neurons in the hidden layer were optimized. 15% data are employed to compute the network generalization and to halt training. When generalization stops improving, the data validation takes place. The remaining 15% data give an independent estimate of the network performance during and after the training, called testing data. But for Bayesian Regularization algorithm assisted training process, the previously taken 15% data for validation is not needed. In this case 70% data was taken for training purpose and rest 30% data was allotted for testing.

Adaptive neuro-fuzzy inference system (ANFIS)

The network framework of the ANFIS is illustrated in Fig. S6†. It consists of five connected layers (excluding input layer) which is common for the two input dimensions, P and Q, both of which possess three fuzzy sets, viz. C1C2C3 for P, while D1D2D3 for Q input. We have chosen A number of inputs and B number of fuzzy set to represent each input which in turn implies A×B number of nodes in Layer1. In Layer 2, all the nodes are interconnected with the membership function output of each input node, yielding a total of B^A node in Layer 2. Layer 3 and 4 possess the same number of nodes as that of Layer 2. Layer 5, on the other hand, possess only one node representing the output of the network. Upon considering each input as a node, the total number of nodes in the architecture will be A + A×B + 3×B^A + 1. In ANFIS, only the membership function parameters in Layer1 and inputs weight in Layer 4

are to be predicted by training. Upon implication of the triangular membership function (*trimf*) which is represented by three parameters, we need to assess 3×B×A premise parameters in Layer 1 and A×B^A consequent weight parameters in Layer 4.

The structure of the ANFIS is automatically tuned by least-squares estimation and the back-propagation algorithm. A fuzzy set A of a universe of discourse X is represented by a collection of ordered pairs of generic elements and its membership function $\mu_A(x)$: *X* tends to [0 1], which associates a number $\mu A(x)$ to each element *x* of *X*. The fuzzy logic controller works on the basis of a set of control rules (called the fuzzy rules) among the linguistic variables. These fuzzy rules are represented in the form of conditional statements.

The basic structure of the pattern predictor model developed using ANFIS to predict the pattern of the flow regime consists of four important parts, namely, the fuzzification, knowledge base, artificial neural network, and defuzzification blocks, as shown in Fig. S7†. The inputs to the ANFIS are the H^+ and OAc^-. These are fed to the fuzzification unit, which converts the binary data into linguistic variables. These in turn are given as inputs to the knowledge base block. The ANFIS tool in MATLAB 2018 developed 25 rules while training the neural network. The knowledge base block is connected to the artificial neural network block. A hybrid optimization algorithm is used to train the neural network and to select the proper set of rules for the knowledge base. To predict the current intensity values at 0.44V and 0.64V, training is an important step in the selection of the proper rule base. Once the proper rule base is selected, the ANFIS model is ready to carry out prediction. The trained ANFIS was validated using 15% of the data. The output of the artificial neural network unit is given as input to the defuzzification unit, where the linguistic variables are converted back into numerical data in crisp form.

Computational details of decision tree regression (DTR)

We have used decision tree regression for the computational prediction of our chemical data using the python programming language. Chemical data followed by its header has been imported, using the ‘pandas’ library. Then the datasets are split into two parts train, and test, using the ‘scikit-learn’ library function 'train_test_split'. Then we fitted the dataset with decision tree regression using the Scikit-learn library function 'DecisionTreeRegressor'. We have got an optimized depth of the tree by calculating training accuracy. We have taken one less depth from the maximum depth corresponding to maximum accuracy to avoid decision tree over fitting. We have plotted the decision tree with the optimized depth of the tree.

Python codes for decision tree regression.

```
#import inspect
import pandas as pd
import matplotlib.pyplot as plt
#work_dir = inspect.currentframe()
#file = inspect.getfile(work_dir)
#path = file.split('.')[0]
# read the dataset / data and programme has to be in same directory
chem_input = pd.read_csv('dataset_mod.csv')
chem_input.shape
# train test data formation
x = chem_input.drop(['0.44V','0.64V' ],axis ='columns') # independent variables
y = chem_input[['0.44V','0.64V']] # dependent variable
#y = chem_input[['0.44V']]
#y = chem_input['0.44V'] # dependent variable
#y = chem_input['0.64V'] # dependent variable
#from sklearn.datasets import make_regression
#xx,yy = make_regression(n_sample = 10, n_features = 2, n_targets = 2)
from skleam.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state = 0,
test_size =0.13)
# varification of data set
y_train.value_counts(normalize =True)
y_test.value_counts(normalize = True)
x_train.shape, y_train.shape
x_test.shape, y_test.shape
# decision tree
```

1

```
from skleam.tree import DecisionTreeRegressor
chem_tree = DecisionTreeRegressor()
chem_tree.fit(x_train,y_train)
# mean accuracy of the data set
t = chem_tree.score(x_train, y_train)
tt = chem_tree.score(x_test,y_test)
# predicted regression value and probability for y_test / for dependent
variable
pred_yt = chem_tree.predict(x_test)
# model evaluation using R square for DTR
from skleam import metrics
r_square = metrics.r2_score(y_test,pred_yt)
#pred_yt_prob =chem_tree.predict_proba(x_test)
# Do optimization by depth as well as by other features like leaf_nodes
train_accuracy =[]
test_accuracy =[]
for d in range(1,11):
    chem_tree = DecisionTreeRegressor(max_depth = d, random_state = 0)
    chem_tree.fit(x_train,y_train)
    train_accuracy.append(chem_tree.score(x_train, y_train))
    test_accuracy.append(chem_tree.score(x_test, y_test))
optimized_dataframe = pd.DataFrame({'max_depth':range(1,11), 'train_acc':
train_accuracy, 'test_acc': test_accuracy})
#optimized_dataframe.head()
plt.figure(figsize=(12,6),dpi=600)
plt.plot(optimized_dataframe['max_depth'], optimized_dataframe['train_acc'],
color = 'green',  marker ='o')
```

2

```
plt.plot(optimized_dataframe['max_depth'], optimized_dataframe['test_acc'],
color = 'red',  marker ='x')
plt.xlabel('Depth of tree')
plt.ylabel('Performance')
plt.legend(['Train_accuracy', 'Test_accuracy'])
#plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.savefig("chem_optimized.png")
# after optimization gives input of that variable
md_test = test_accuracy.index(max(test_accuracy))
md_train = train_accuracy.index(max(train_accuracy))
max_depth_opt = md_train + 1

chem_tree_opt = DecisionTreeRegressor(max_depth =
max_depth_opt,random_state = 10)
chem_tree_opt.fit(x_train,y_train)
opt_train = chem_tree_opt.score(x_train, y_train)
opt_test = chem_tree_opt.score(x_test,y_test)
pred_yt_opt = chem_tree_opt.predict(x_test)
rmse_opt = chem_tree_opt.score(x_test,y_test)
#r_square_opt = metrics.r2_score(y_test,pred_yt_opt)
# plot tree
from skleam import tree
#fig, axes = plt.subplots(nrows = 1, ncols = 1, figsize = (25,20), dpi = 600)
plt.figure(figsize =(15,10), dpi =800)
decision_tree_plot = tree.plot_tree(chem_tree, feature_names =
x_train.columns, max_depth= max_depth_opt,filled=True)
#fig.savefig("chem_tree.png")plt.savefig("C:/Users/anik0/OneDrive/Desk
top/Sohini_Manuscript/chem_tree.png")
```

3

Tables for electronic supplementary information

Table S1. Rules for the fuzzy logic system (Mamdani based) by taking H^+ (input 1) and OAc^- (input 2) as the inputs and current intensity at 0.44V and 0.64V as the outputs. The rules consist of the following statements.

1. If (input1 is L) then (output1 is L)(output2 is H) (1)
2. If (input1 is M) then (output1 is L)(output2 is H) (1)
3. If (input1 is H) then (output1 is L)(output2 is H) (1)
4. If (input2 is L) then (output1 is L)(output2 is H) (1)
5. If (input2 is M) then (output1 is M)(output2 is M) (1)
6. If (input2 is H) then (output1 is H)(output2 is L) (1)
7. If (input1 is L) and (input2 is L) then (output1 is L)(output2 is H) (1)
8. If (input1 is M) and (input2 is L) then (output1 is L)(output2 is H) (1)
9. If (input1 is H) and (input2 is L) then (output1 is L)(output2 is H) (1)
10. If (input1 is L) and (input2 is M) then (output1 is M)(output2 is M) (1)
11. If (input1 is M) and (input2 is M) then (output1 is M)(output2 is M) (1)
12. If (input1 is H) and (input2 is M) then (output1 is M)(output2 is M) (1)
13. If (input1 is L) and (input2 is H) then (output1 is H)(output2 is L) (1)
14. If (input1 is M) and (input2 is H) then (output1 is M)(output2 is M) (1)
15. If (input1 is H) and (input2 is H) then (output1 is M)(output2 is M) (1)

Table S2. Values of current intensity as a function of n_{H^+}/n_1 and n_{OAc^-}/n_1.

No.	Equv. of H^+	Equv. of OAc^-	Current Intensity at 0.44V	Current Intensity at 0.64V
1	2.9	0.008	0.01	13.99
2	2.75	0.076	0.99	12.73
3	2.67	0.12	1.42	12.19
4	2.46	0.13	1.77	11.92
5	2.28	0.15	1.92	11.78
6	2.21	0.16	3.43	11.7
7	2.08	0.21	3.65	10.13
8	1.98	0.27	4.39	9.16
9	1.79	0.29	6.73	8.94
10	1.66	0.3	4.47	8.87
11	1.59	0.38	6.73	6.5
12	1.45	0.43	6.8	6.35
13	1.38	0.46	7.31	6.28
14	1.25	0.54	8.25	5.16
15	1.04	0.6	9.8	3.94
16	1.02	0.62	10.02	3.74
17	1	0.69	10.26	2.87
18	0.99	0.73	11.5	2.64
19	0.91	0.77	11.65	2.42
20	0.88	0.89	13.69	1.45
21	0.85	0.9	14.01	1.32
22	0.75	0.92	14.69	1.25
23	0.72	0.98	14.9	1.11
24	0.67	1.01	15.01	1.04
25	0.63	1.04	15.13	0.95
26	0.57	1.07	15.22	0.86
27	0.51	1.29	15.3	0.82
28	0.43	1.4	15.37	0.81
29	0.41	1.49	15.45	0.79
30	0.35	1.55	15.56	0.75
31	0.31	1.7	15.67	0.1
32	0.25	1.77	15.68	0.59
33	0.21	1.99	15.74	0.53
34	0.15	2.03	15.79	0.45
35	0.12	2.1	15.81	0.39
36	0.09	2.18	15.88	0.31
37	0.06	2.3	16	0.27
38	0.02	2.69	16.05	0.15
39	0.018	2.73	16.11	0.1
40	0.009	2.89	16.2	0.09

Table S3. Rules for the fuzzy logic system (based on Sugeno's Method) by taking H^+ as input 1 and OAc^- as input 2, whereas current intensity at 0.44V as the output. The rules consist of the following statements.

1. If (input1 is in1mf1) and (input2 is in2mf1) then (output is out1mf1) (1)
2. If (input1 is in1mf1) and (input2 is in2mf2) then (output is out1mf2) (1)
3. If (input1 is in1mf1) and (input2 is in2mf3) then (output is out1mf3) (1)
4. If (input1 is in1mf1) and (input2 is in2mf4) then (output is out1mf4) (1)
5. If (input1 is in1mf1) and (input2 is in2mf5) then (output is out1mf5) (1)
6. If (input1 is in1mf2) and (input2 is in2mf1) then (output is out1mf6) (1)
7. If (input1 is in1mf2) and (input2 is in2mf2) then (output is out1mf7) (1)
8. If (input1 is in1mf2) and (input2 is in2mf3) then (output is out1mf8) (1)
9. If (input1 is in1mf2) and (input2 is in2mf4) then (output is out1mf9) (1)
10. If (input1 is in1mf2) and (input2 is in2mf5) then (output is out1mf10) (1)
11. If (input1 is in1mf3) and (input2 is in2mf1) then (output is out1mf11) (1)
12. If (input1 is in1mf3) and (input2 is in2mf2) then (output is out1mf12) (1)
13. If (input1 is in1mf3) and (input2 is in2mf3) then (output is out1mf13) (1)
14. If (input1 is in1mf3) and (input2 is in2mf4) then (output is out1mf14) (1)
15. If (input1 is in1mf3) and (input2 is in2mf5) then (output is out1mf15) (1)
16. If (input1 is in1mf4) and (input2 is in2mf1) then (output is out1mf16) (1)
17. If (input1 is in1mf4) and (input2 is in2mf2) then (output is out1mf17) (1)
18. If (input1 is in1mf4) and (input2 is in2mf3) then (output is out1mf18) (1)
19. If (input1 is in1mf4) and (input2 is in2mf4) then (output is out1mf19) (1)
20. If (input1 is in1mf4) and (input2 is in2mf5) then (output is out1mf20) (1)
21. If (input1 is in1mf5) and (input2 is in2mf1) then (output is out1mf21) (1)
22. If (input1 is in1mf5) and (input2 is in2mf2) then (output is out1mf22) (1)
23. If (input1 is in1mf5) and (input2 is in2mf3) then (output is out1mf23) (1)
24. If (input1 is in1mf5) and (input2 is in2mf4) then (output is out1mf24) (1)
25. If (input1 is in1mf5) and (input2 is in2mf5) then (output is out1mf25) (1)

Table S4. Rules for the fuzzy logic system (based on Sugeno's method) by taking input 1 (H^+) and input 2 (OAc^-) as the inputs and current intensity at 0.64V the output. The rules consist of the following statements.

1. If (input1 is in1mf1) and (input2 is in2mf1) then (output is out1mf1) (1)
2. If (input1 is in1mf1) and (input2 is in2mf2) then (output is out1mf2) (1)
3. If (input1 is in1mf1) and (input2 is in2mf3) then (output is out1mf3) (1)
4. If (input1 is in1mf1) and (input2 is in2mf4) then (output is out1mf4) (1)
5. If (input1 is in1mf1) and (input2 is in2mf5) then (output is out1mf5) (1)
6. If (input1 is in1mf2) and (input2 is in2mf1) then (output is out1mf6) (1)
7. If (input1 is in1mf2) and (input2 is in2mf2) then (output is out1mf7) (1)
8. If (input1 is in1mf2) and (input2 is in2mf3) then (output is out1mf8) (1)
9. If (input1 is in1mf2) and (input2 is in2mf4) then (output is out1mf9) (1)
10. If (input1 is in1mf2) and (input2 is in2mf5) then (output is out1mf10) (1)
11. If (input1 is in1mf3) and (input2 is in2mf1) then (output is out1mf11) (1)
12. If (input1 is in1mf3) and (input2 is in2mf2) then (output is out1mf12) (1)
13. If (input1 is in1mf3) and (input2 is in2mf3) then (output is out1mf13) (1)
14. If (input1 is in1mf3) and (input2 is in2mf4) then (output is out1mf14) (1)
15. If (input1 is in1mf3) and (input2 is in2mf5) then (output is out1mf15) (1)
16. If (input1 is in1mf4) and (input2 is in2mf1) then (output is out1mf16) (1)
17. If (input1 is in1mf4) and (input2 is in2mf2) then (output is out1mf17) (1)
18. If (input1 is in1mf4) and (input2 is in2mf3) then (output is out1mf18) (1)
19. If (input1 is in1mf4) and (input2 is in2mf4) then (output is out1mf19) (1)
20. If (input1 is in1mf4) and (input2 is in2mf5) then (output is out1mf20) (1)
21. If (input1 is in1mf5) and (input2 is in2mf1) then (output is out1mf21) (1)
22. If (input1 is in1mf5) and (input2 is in2mf2) then (output is out1mf22) (1)
23. If (input1 is in1mf5) and (input2 is in2mf3) then (output is out1mf23) (1)
24. If (input1 is in1mf5) and (input2 is in2mf4) then (output is out1mf24) (1)
25. If (input1 is in1mf5) and (input2 is in2mf5) then (output is out1mf25) (1)

Figures for electronic supplementary information

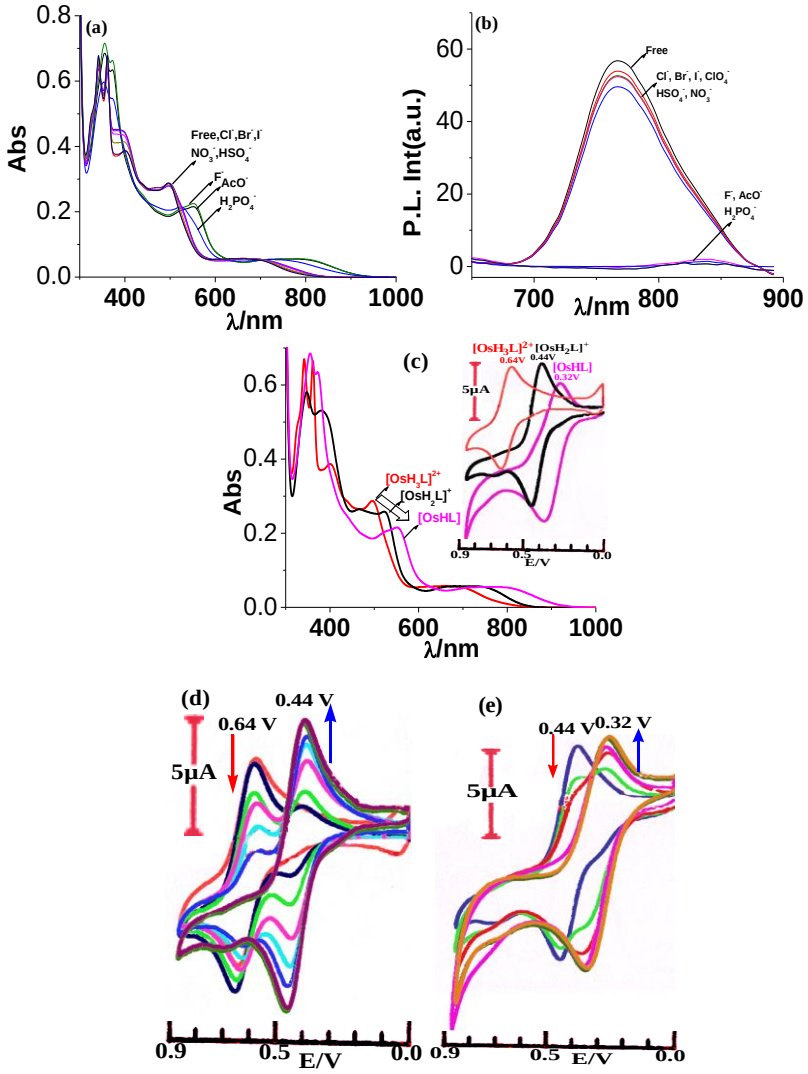


Fig. S1 Change of UV-vis absorption (a) and emission (b) spectrum of **1** in CH_2Cl_2 upon addition of different anions as TBA salts. (c) UV-vis spectra and cyclic voltammograms of **1** in three different protonated states $[(bpy)_2Os(H_3Imbzim)]^{2+}$, $[(bpy)_2Os(H_2Imbzim)]^{+}$, and $[(bpy)_2Os$ (HImbzim)]. (d) and (e) represent CVs of receptor **1** obtained upon incremental addition of TBOAc to its acetonitrile solution (1.0×10^{-3} M) and the changes in the current intensities as a function of equivalents of OAc^- ion added.

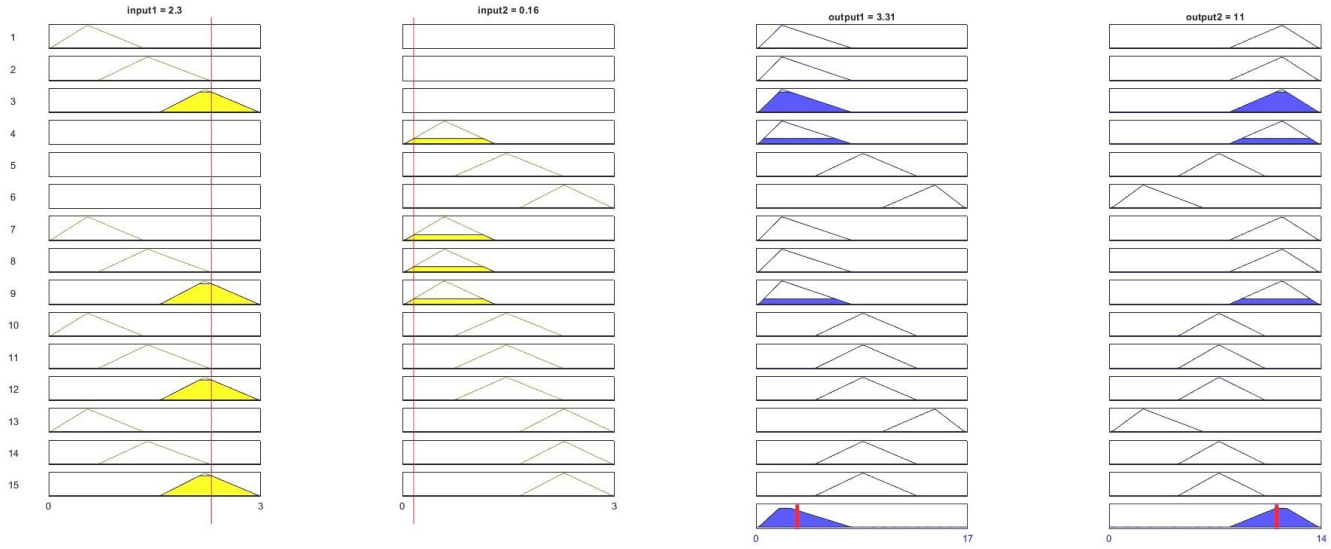


Fig. S2 Mamdani rule view for **1.**

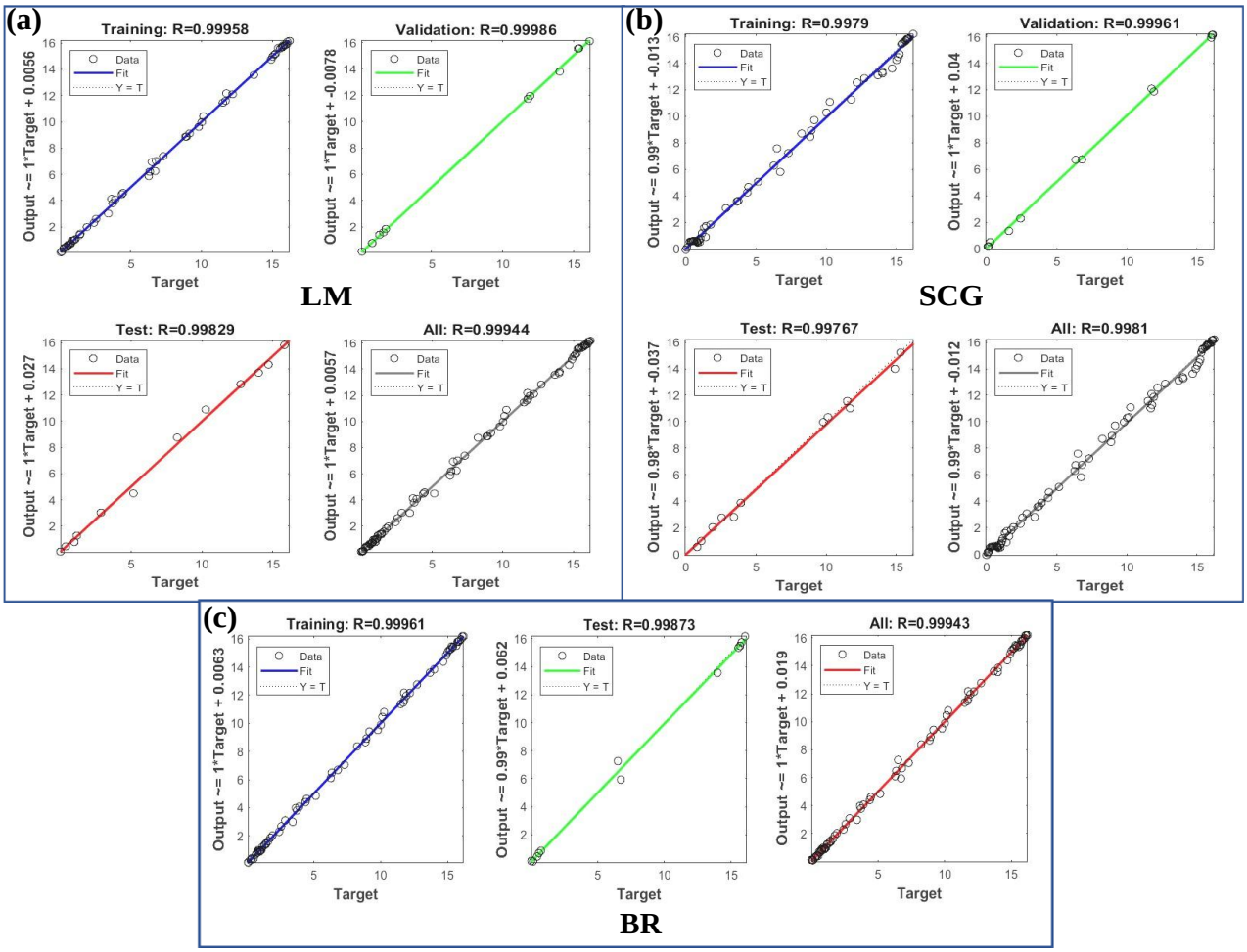


Fig. S3 The outcomes of ANN models via linear regression (a-c) at 0.44V and 0.64V respectively.

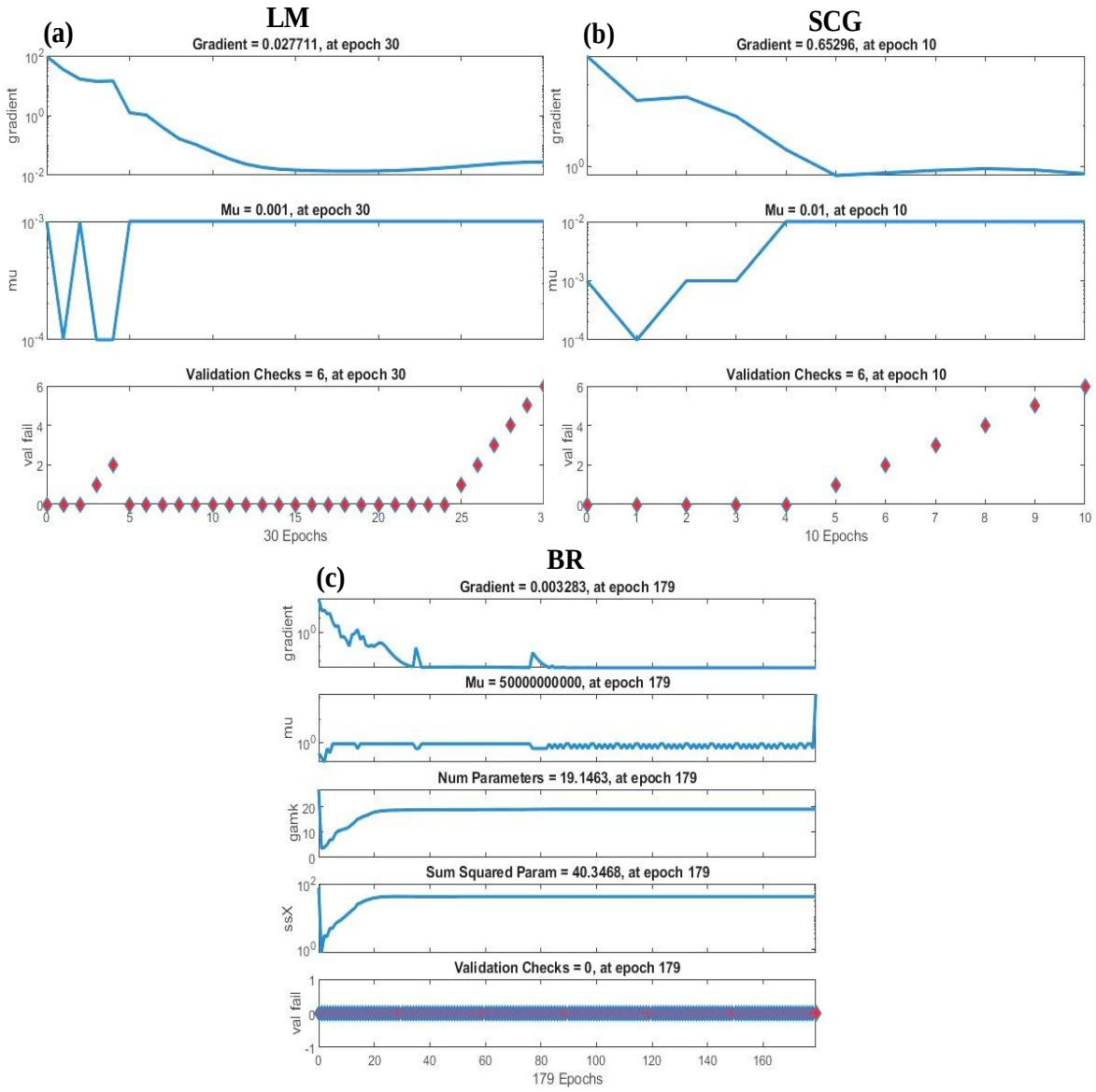


Fig. S4 Training states of Levenberg-Marquardt (LM) (a), Scaled Conjugate Gradient (SCG) (b) and Bayesian-Regularization (BR) (c) algorithms.

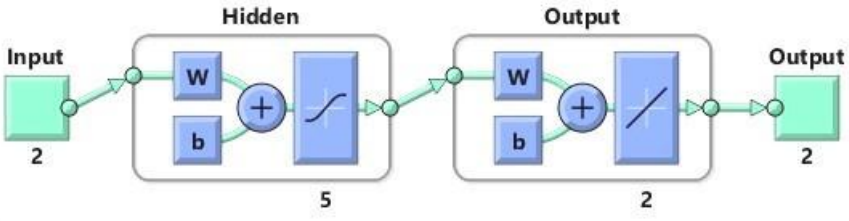


Fig. S5 Schematic presentation of ANN consisting of 2 inputs, 5 hidden layers and 2 outputs.

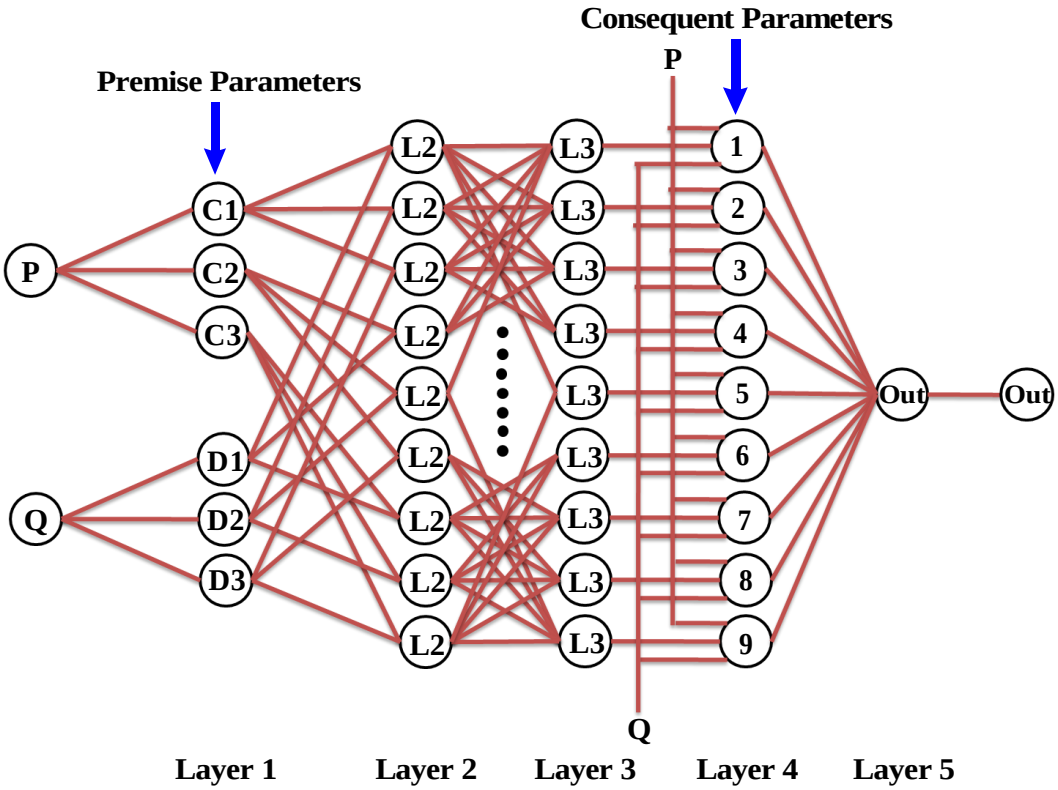


Fig. S6 Schematic sketch of ANFIS network comprising of two inputs, five layers and one output.

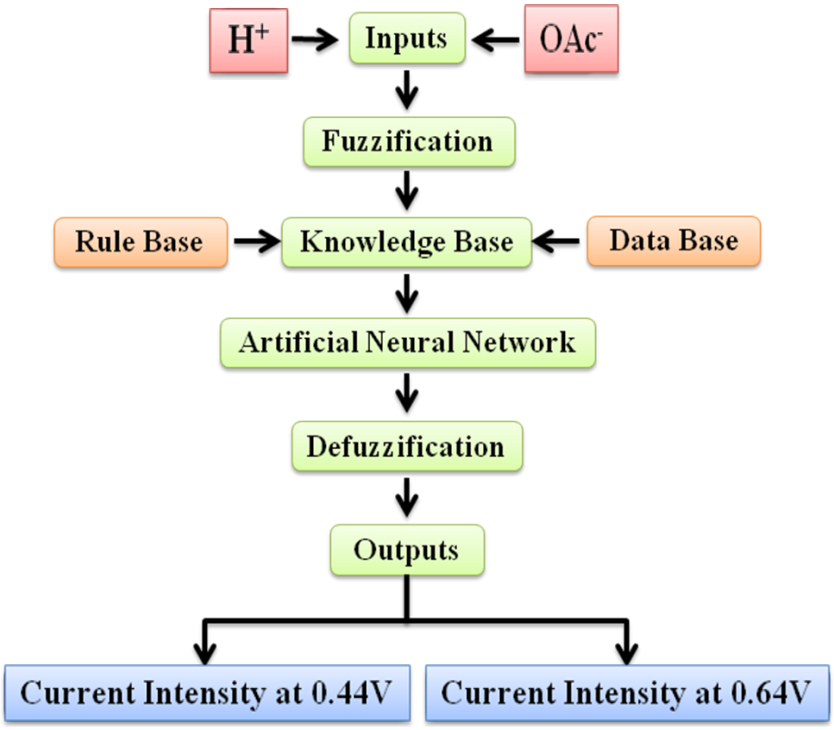


Fig. S7 Block diagram of the ANFIS for predicting the output in presence of inputs.

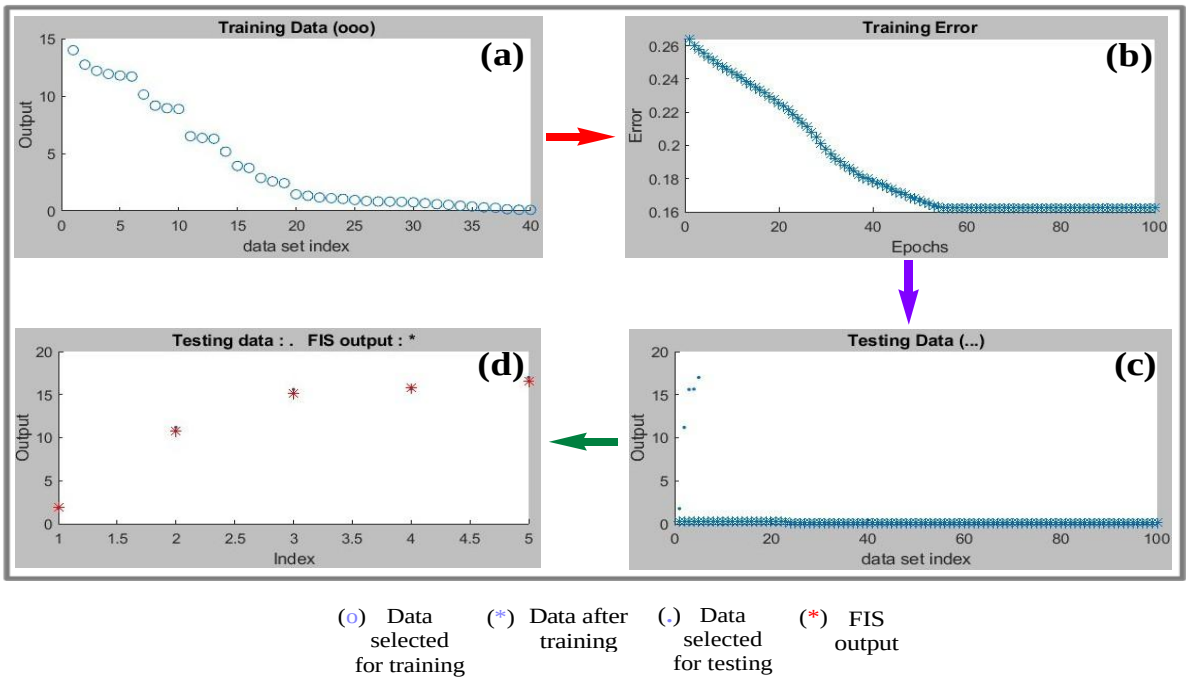


Fig. S8 (a) Specific dataset for training the ANFIS network. (b) Progressive decrease of error in each training steps up to 100 epochs. (c) Data to test the trained model accuracy. (d) Comparison of testing data with fuzzy inference system outputs.

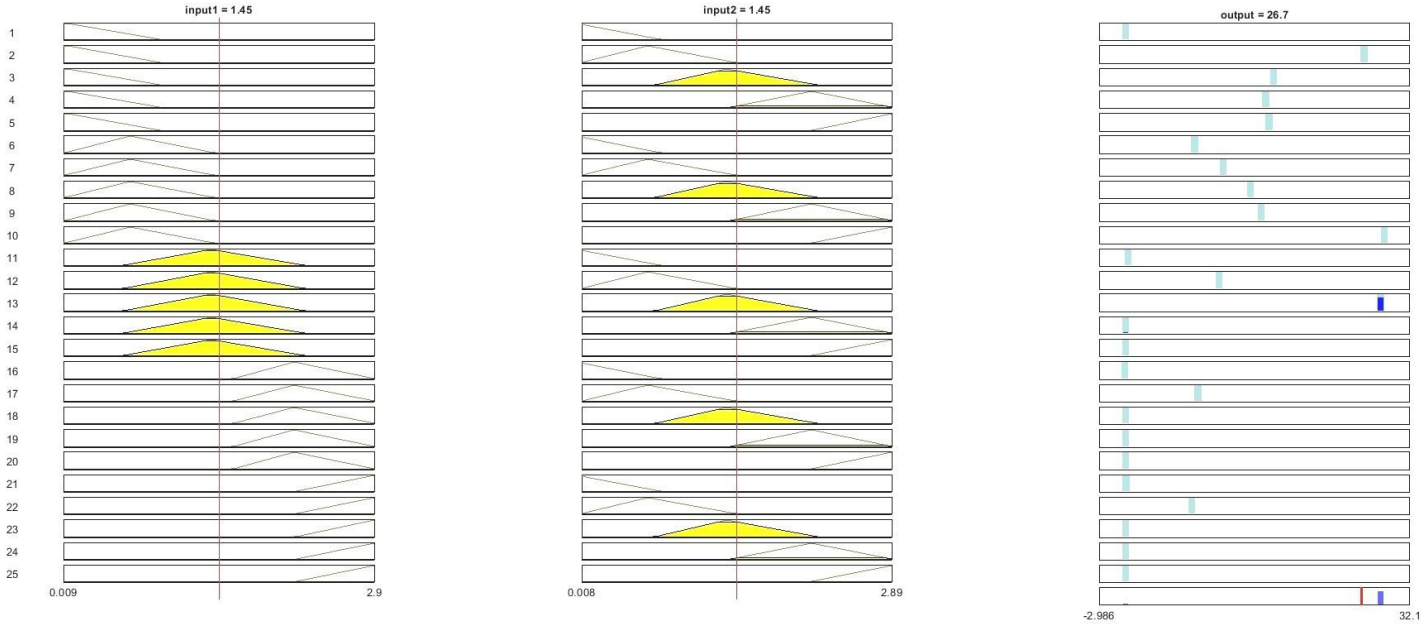


Fig. S9 Sugeno rule view (output at 0.44V) for **1**.

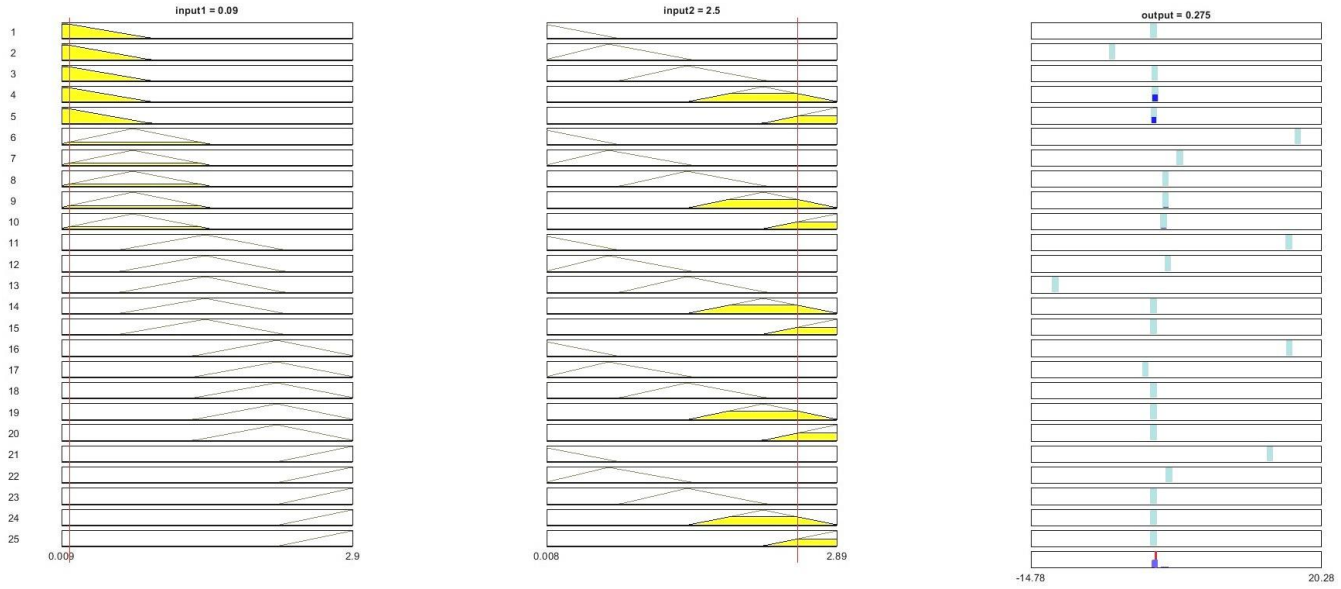


Fig. S10 Sugeno rule view (output at 0.64V) for **1**.

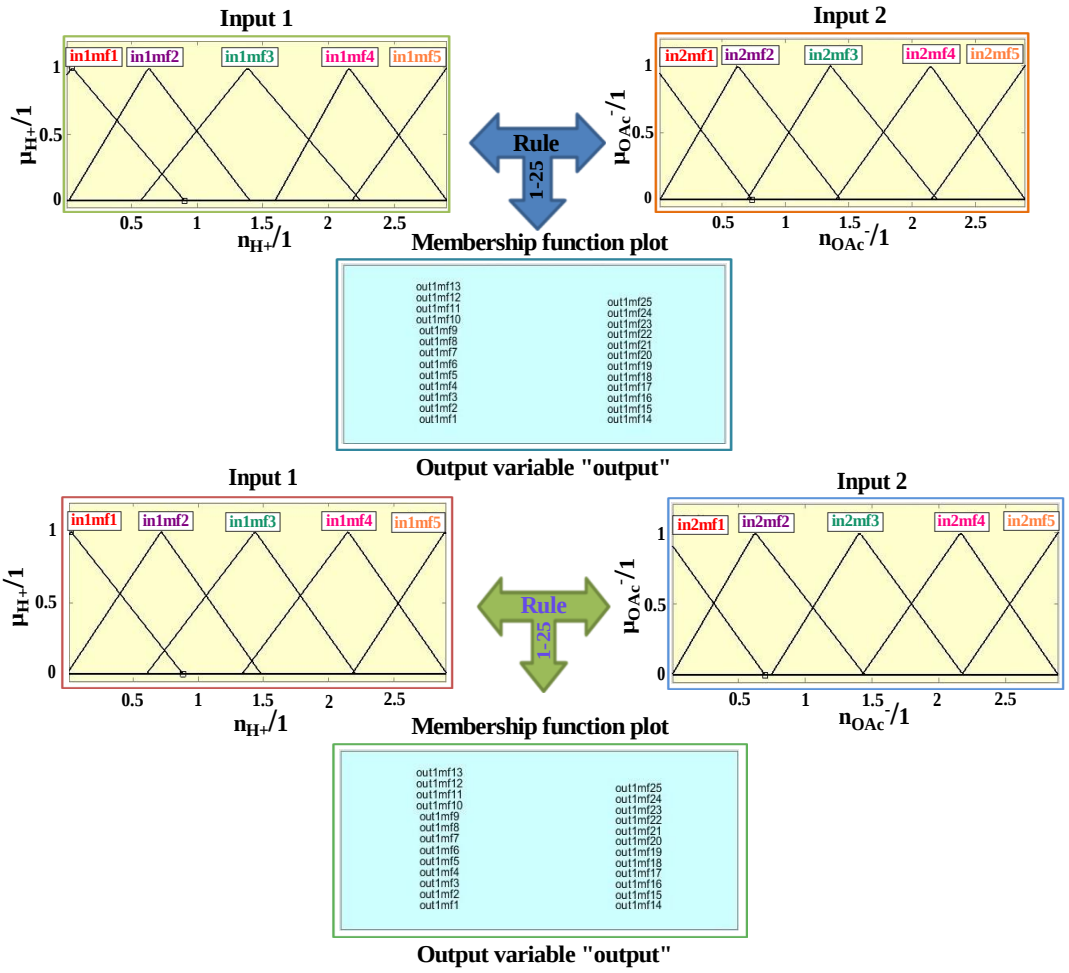


Fig. S11 Range of the five triangular membership (trimf) functions of each inputs (H^+ and OAc^-) and 25 rules based ANFIS generated 25 output (at 0.44V and 0.64V) membership functions.

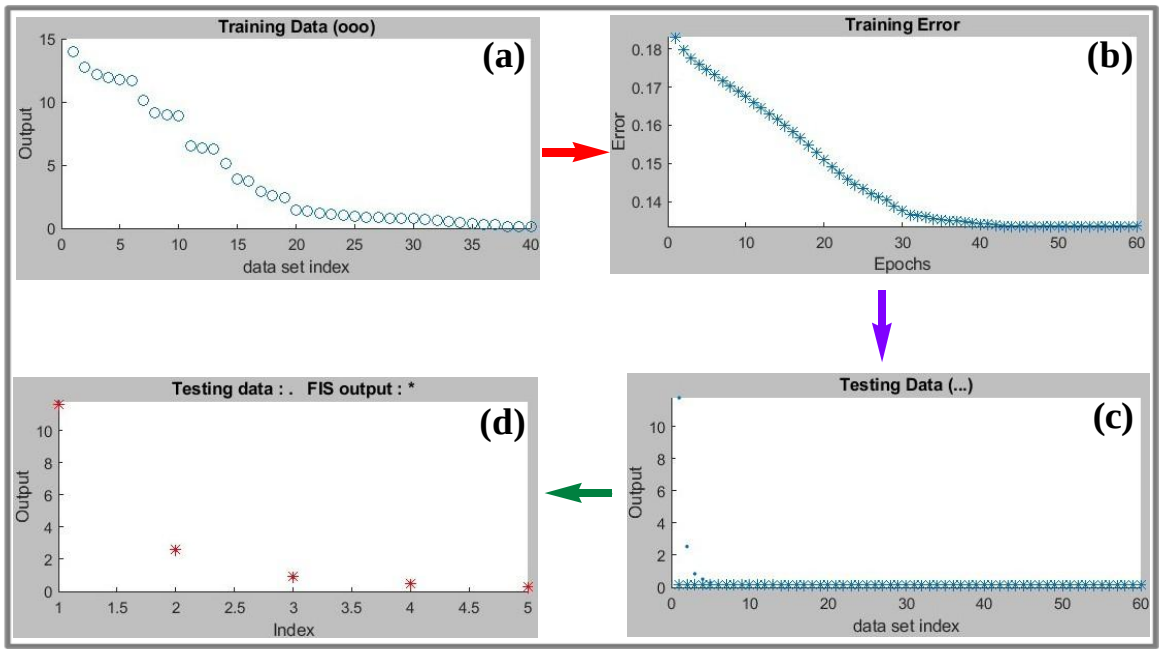


Fig. S12 (a) Selected training data to design the ANFIS model (monitoring 0.64V). (b) Training error minimization up to 60 epochs. (c) Combination of training and testing data. (d) Compilation of testing data and FIS output.

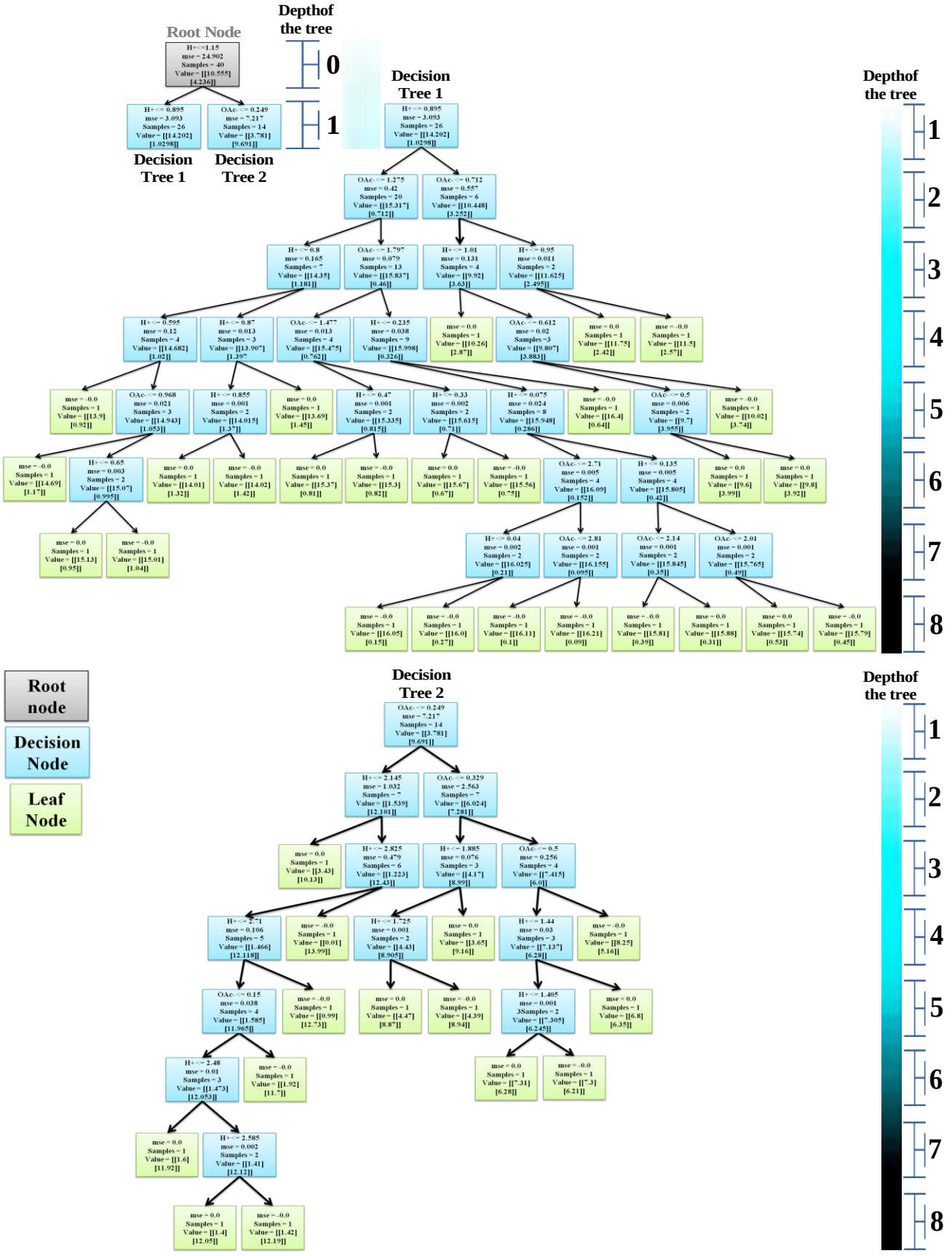
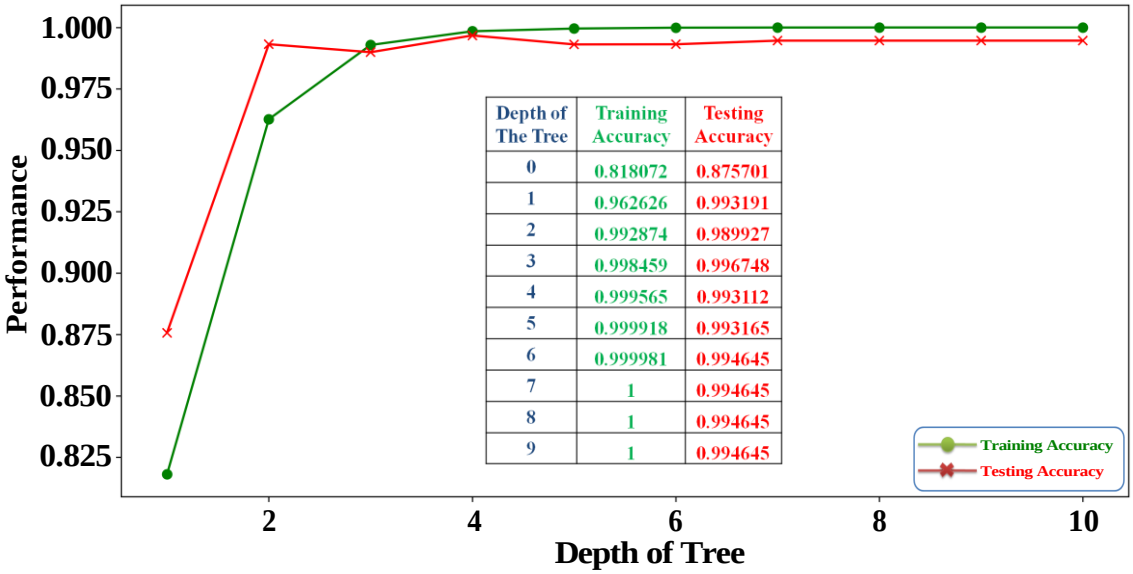


Fig. S13 Representation of the decision tree.



Depth of The Tree	Training Accuracy	Testing Accuracy
0	0.818072	0.875701
1	0.962626	0.993191
2	0.992874	0.989927
3	0.998459	0.996748
4	0.999565	0.993112
5	0.999918	0.993165
6	0.999981	0.994645
7	1	0.994645
8	1	0.994645
9	1	0.994645

Fig. S14 Accomplishment of the DTR up to depth 9.