Electronic Supplementary Material (ESI) for Dalton Transactions.
This journal is © The Royal Society of Chemistry 2022

Electronic Supplementary Information

Neural network and decision tree based machine learning tools for analysis of anion-responsive behaviours of emissive Ru(II)-terpyridine complexes

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

[a] Department of Chemistry, Inorganic Chemistry Section, Jadavpur University, Kolkata-700032, India

[b] School of Biological Sciences, Indian Association for the Cultivation of Science, Kolkata -700 032, India

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 1 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 Figure S7. 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.). 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 S1† 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 40×1 matrix represents the static output data (at 0.48V and 1.11V) of one element. Now, the 40 samples are divided into 3 sets of data. 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 (Fig. 6a and 6b).

Adaptive Neuro-Fuzzy Inference System (ANFIS). The network framework of the ANFIS is illustrated in Figure S8. 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 Figure S9. The inputs to the ANFIS are the H^+ and AcO^-. 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 9 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 0.48V and 1.11V, 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, viz. train and test, using the ‘scikit-learn’ library function 'train_test_split'. Thereafter, 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 the 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.

```
# Import panda and matplotlib
import pandas as pd
import matplotlib.pyplot as plt
# Read the dataset
chem_input = pd.read_csv('Dataset_new_modd.csv')
chem_input.shape
# Train test data formation
from sklearn.model_selection import train_test_split
x = chem_input.drop(['IV_0.48V','IV_1.11V' ],axis ='columns') # independent variables
y = chem_input[['IV_0.48V','IV_1.11V']] # dependent variable
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 Regressor
from sklearn.tree import DecisionTreeRegressor
chem_tree = DecisionTreeRegressor()
chem_tree.fit(x_train,y_train) # initial fitting of the dataset
# Mean accuracy of the data set
TrainMC = chem_tree.score(x_train, y_train)
TestMC = chem_tree.score(x_test,y_test)
# Predicted regression value and probability for y_test, the dependent variable
pred_yt = chem_tree.predict(x_test)
# Model evaluation using R square for DTR
from sklearn import metrics
r_square = metrics.r2_score(y_test,pred_yt)
# Optimization by Depth of the tree 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})
# Plot the DT performance wrt the depth
plt.figure(figsize=(12,6),dpi=600)
plt.plot(optimized_dataframe['max_depth'], optimized_dataframe['train_acc'], color = 'green',
marker ='o')
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.savefig("chem_optimized.png")
# Find Max depth with Max performance/ accuracy
md_test = test_accuracy.index(max(test_accuracy))
md_train = train_accuracy.index(max(train_accuracy))
max_depth_opt = md_train
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)
# Plot Decision Tree
from sklearn 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)
plt.savefig("chem_tree.png")
```

Table S1. Values of current intensity as a function of n_{H^+}/n_1 and n_{AcO^-}/n_1.

Input 1(H+)	Input 2(OAc-)	IV_0.48V	IV_1.11V
2.9	0.01	0.01	3.6
2.85	0.12	0.07	3.49
2.81	0.22	0.15	3.39
2.75	0.29	0.19	3.29
2.68	0.37	0.25	3.01
2.5	0.42	0.3	2.99
2.52	0.45	0.42	2.9
2.5	0.47	0.45	2.69
2.43	0.48	0.47	2.52
2.4	0.51	0.5	2.32
2.33	0.54	0.57	2.11
2.29	0.57	0.7	1.87
2.15	0.59	0.88	1.73
1.98	0.66	0.9	1.63
1.55	0.68	0.95	1.23
1.42	0.8	1.12	1.38
1.38	0.83	1.51	0.94
1.3	0.91	1.62	0.84
1.25	0.95	1.76	0.75
1.21	0.98	2	0.72
1.15	1.08	2.12	0.54
1.05	1.16	2.49	0.5
0.97	1.27	2.61	0.48
0.84	1.4	2.72	0.44
0.79	1.44	2.85	0.37
0.75	1.56	2.97	0.35
0.64	1.62	3.16	0.33
0.55	1.75	3.23	0.31
0.51	1.81	3.34	0.27
0.44	1.97	3.41	0.25
0.42	2.06	3.52	0.24
0.4	2.19	3.59	0.21
0.39	2.25	3.62	0.18
0.35	2.41	3.7	0.16
0.32	2.44	3.705	0.14
0.3	2.57	3.72	0.12
0.25	2.6	3.73	0.1
0.21	2.72	3.75	0.09
0.12	2.83	3.78	0.05
0.05	3	3.8	0.03

Table S2. 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 0.48V 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 in1mf2) and (input2 is in2mf1) then (output is out1mf4) (1)
5. If (input1 is in1mf2) and (input2 is in2mf2) then (output is out1mf5) (1)
6. If (input1 is in1mf2) and (input2 is in2mf3) then (output is out1mf6) (1)
7. If (input1 is in1mf3) and (input2 is in2mf1) then (output is out1mf7) (1)
8. If (input1 is in1mf3) and (input2 is in2mf2) then (output is out1mf8) (1)
9. If (input1 is in1mf3) and (input2 is in2mf3) then (output is out1mf9) (1)

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 1.11V 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 in1mf2) and (input2 is in2mf1) then (output is out1mf4) (1)
5. If (input1 is in1mf2) and (input2 is in2mf2) then (output is out1mf5) (1)
6. If (input1 is in1mf2) and (input2 is in2mf3) then (output is out1mf6) (1)
7. If (input1 is in1mf3) and (input2 is in2mf1) then (output is out1mf7) (1)
8. If (input1 is in1mf3) and (input2 is in2mf2) then (output is out1mf8) (1)
9. If (input1 is in1mf3) and (input2 is in2mf3) then (output is out1mf9) (1)

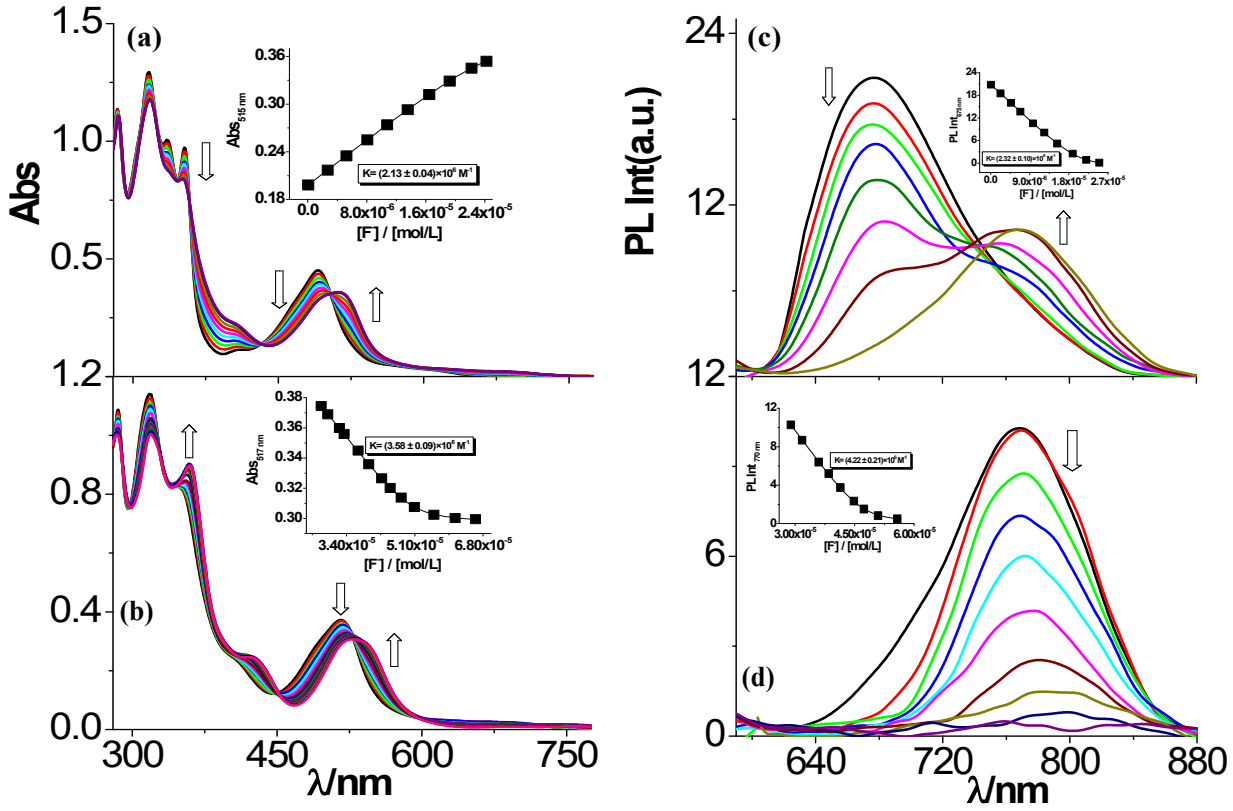

Fig. S1 Changes in absorption [(a) 0–1 equiv. and (b) 1–4 equiv.] and photoluminescence [(c) 0–1 equiv. and (d) 1–4 equiv.] spectra of **1** in acetonitrile-dichloromethane (1:9 v/v) solution (2.0×10^{-5} M) upon incremental addition of F^- ion (5.0×10^{-3} M). The insets show the fit of the experimental absorbance (a and b) and luminescence (c and d) data to a 1:1 binding profile.

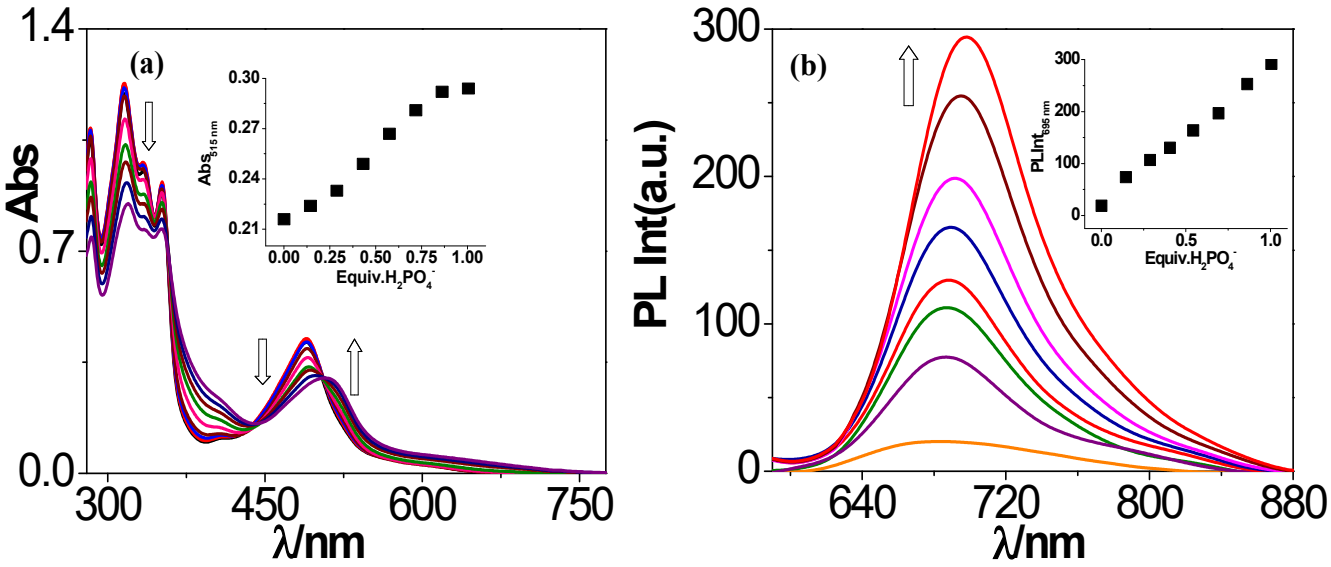

Fig. S2 Changes in absorption (a) and photoluminescence (b) spectra of **1** in acetonitrile-dichloromethane (1:9 v/v) solution (2.0×10^{-5} M) upon the addition of $H_2PO_4^-$ ion (5.0×10^{-3} M). The insets show the change of absorbance (a) and luminescence intensity (b) as a function of the equivalent of $H_2PO_4^-$ ion.

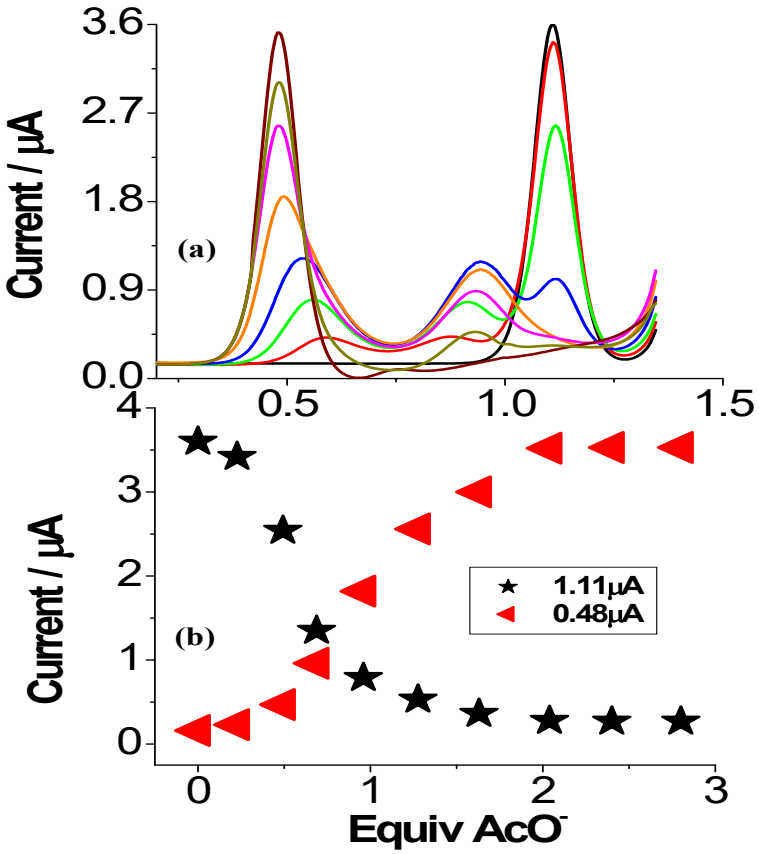

Fig. S3 SWVs (a) of **1** obtained upon incremental addition of AcO^- ion (2.0×10^{-2} M) to its acetonitrile solution (2.5×10^{-4} M). The changes in the current intensities as a function of equivalents of AcO^- ion added are shown in (b).

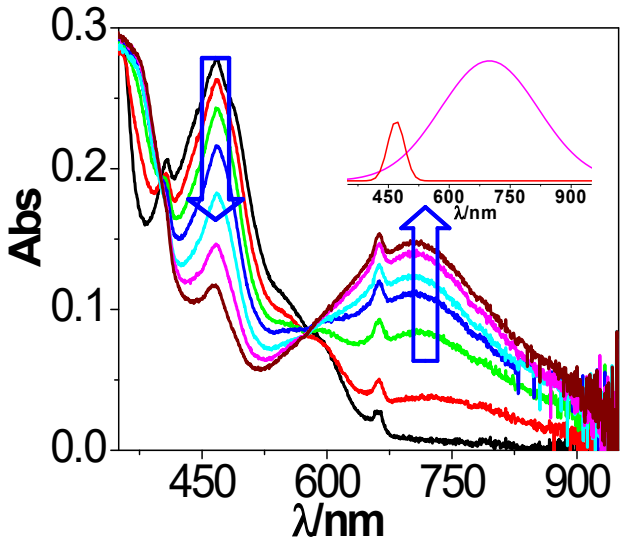

Fig. S4 Spectroelectrochemical changes during the oxidation of **1** and the inset shows the deconvoluted MLCT and LMCT band.

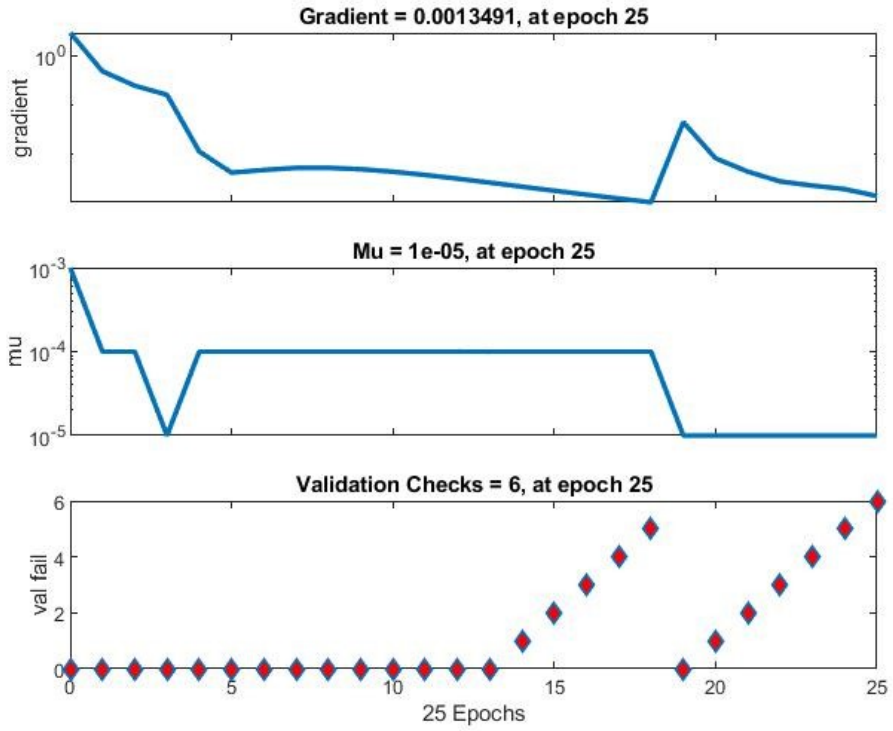

Fig. S5 Training state of the ANN model of **2** at (0.48V) up to epoch 19.

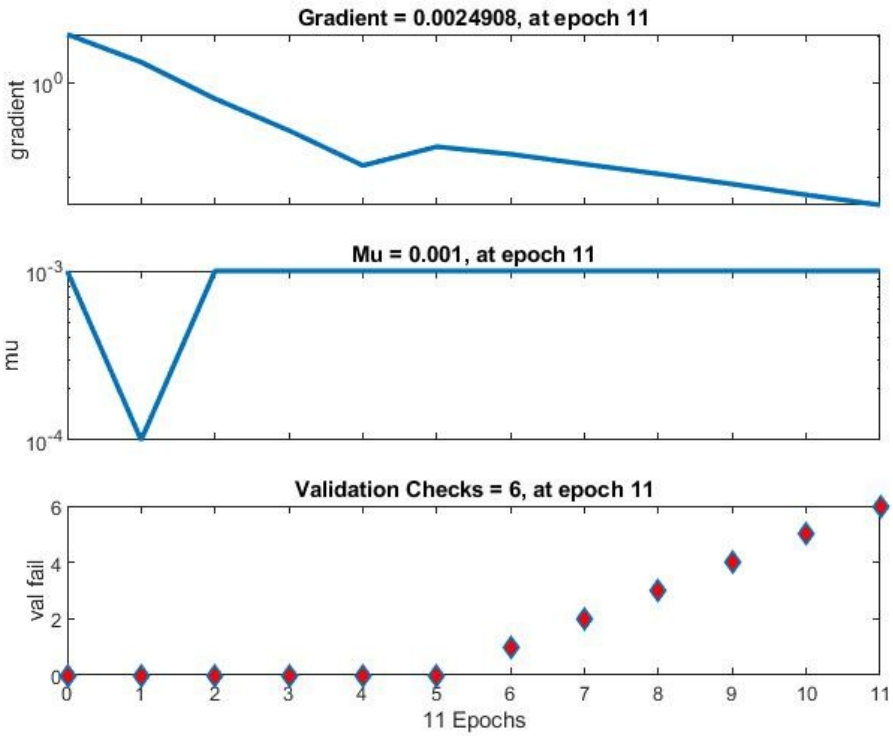

Fig. S6 Training state of the ANN model of **2** at (1.11V) up to epoch 5.

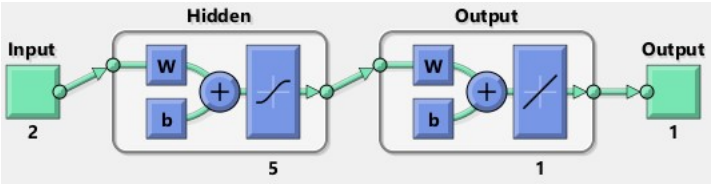

Fig. S7 Artificial neural network model consisting of 2 inputs, 5 hidden layers and 1 output.

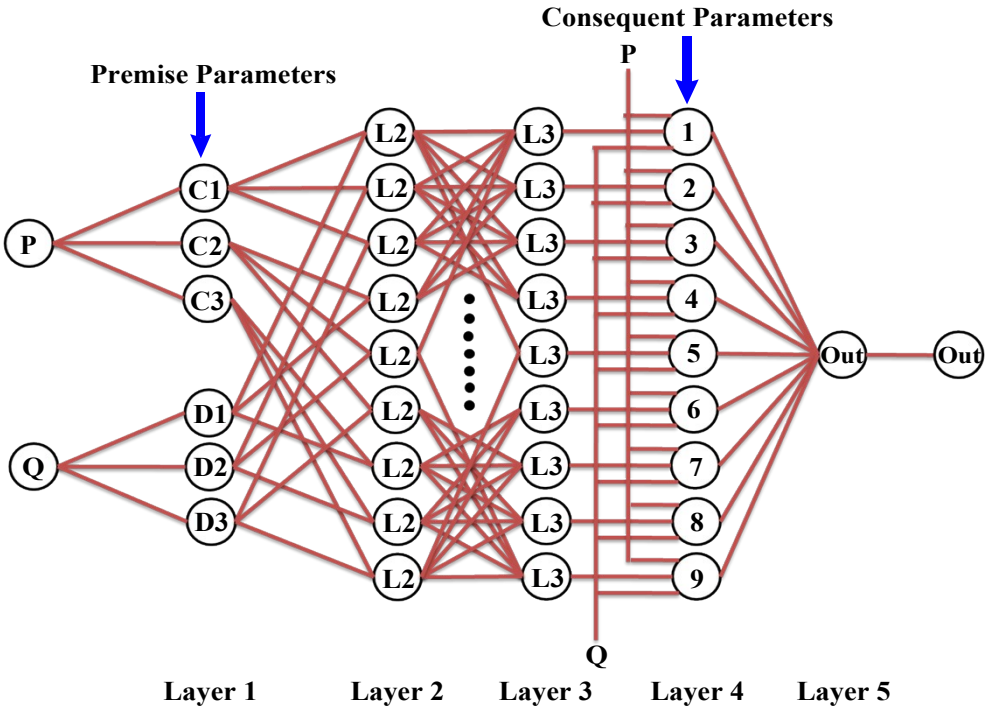

Fig. S8 Schematic sketch of ANFIS network comprising of two inputs, five layers and one output.

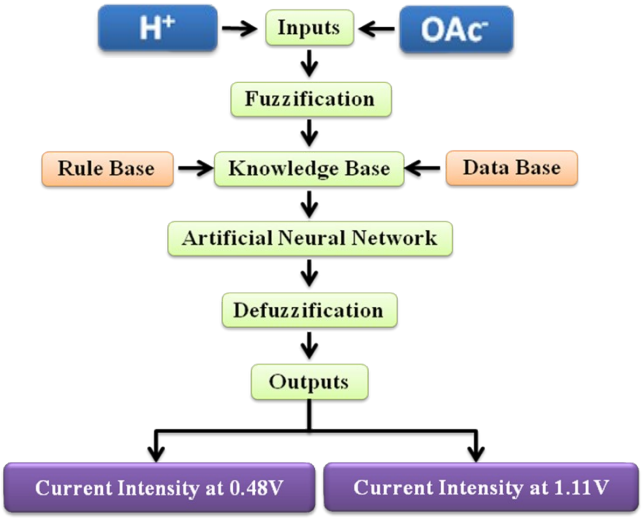

Fig. S9 Block diagram of the ANFIS for predicting the output in presence of inputs.

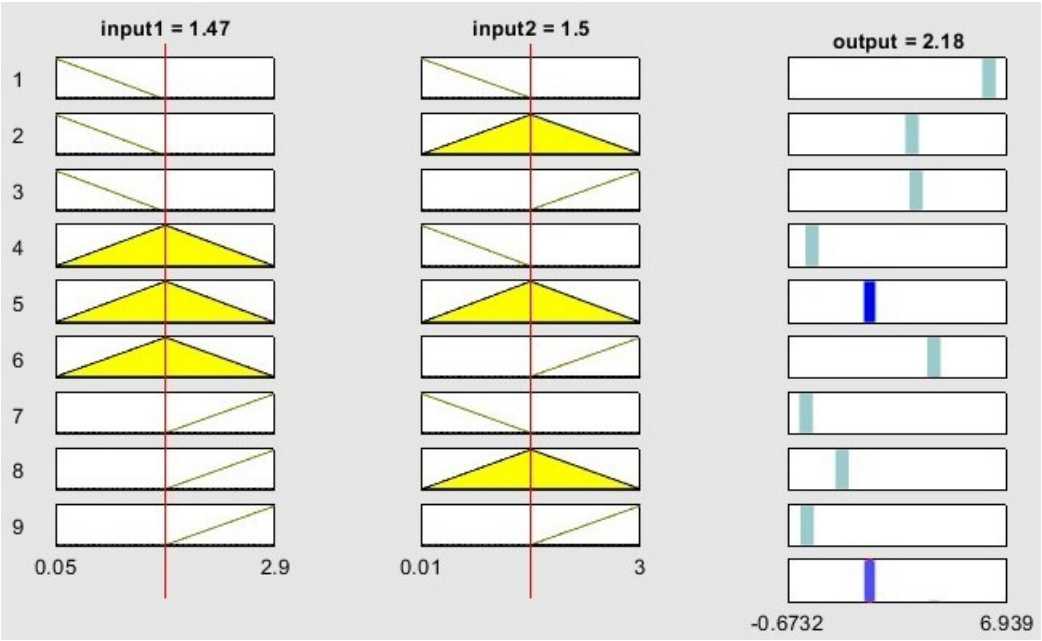

Fig. S10 Sugeno rule view (output at 0.48V) for **2**.

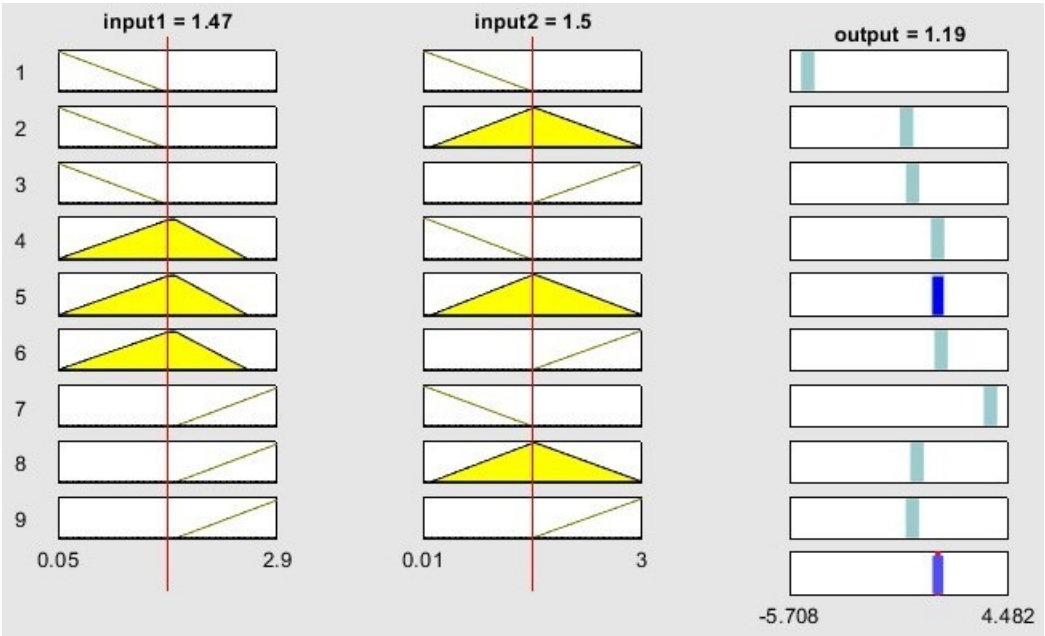

Fig. S11 Sugeno rule view (output at 1.11V) for **2**.

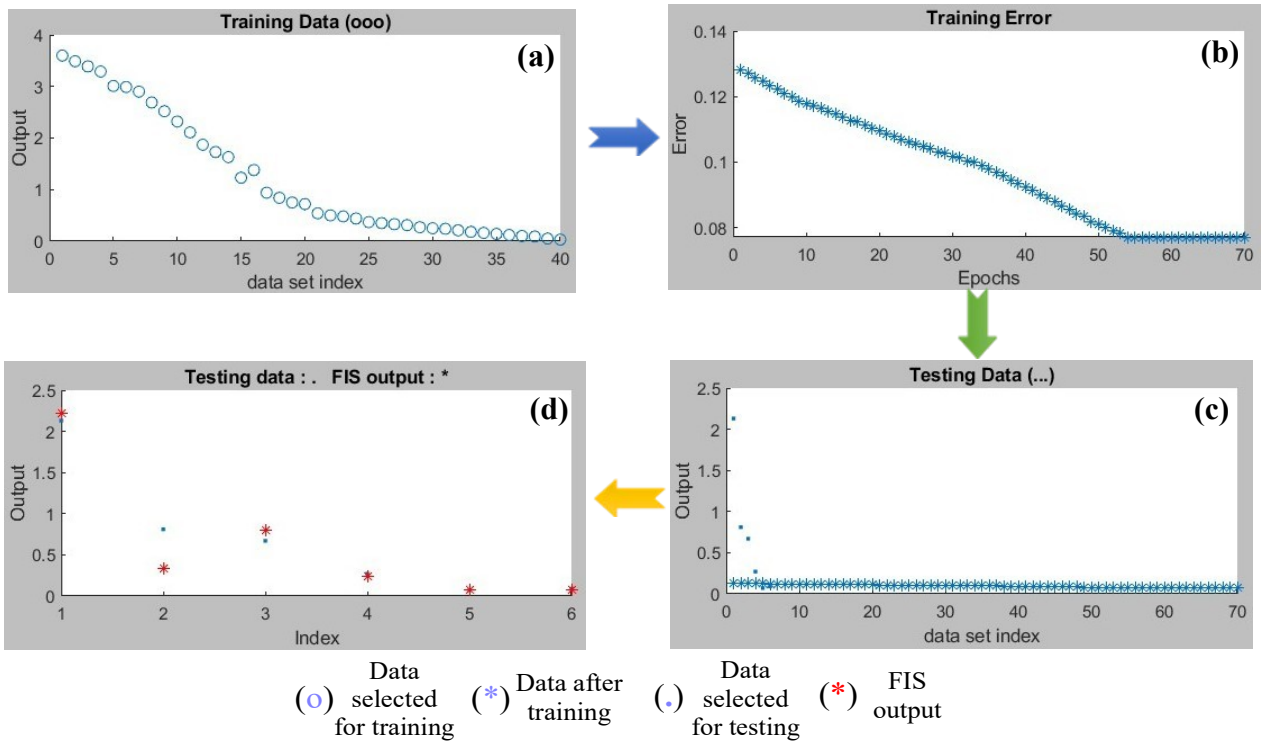

Fig. S12 (a) Selected training data to design the ANFIS model (monitoring 1.11V). (b) Training error minimization up to 100 epochs. (c) Compilation of training set and testing data. (d) Compilation of testing data and FIS output.