Intelligent Systems - Assignment 2 - Neural Fuzzy

Francisco Carvalho | Nº 111000

Wisconsin Breast Cancer Original Dataset

Loading dataset and Creating train-test sets

```
clear all
cd DATA
breast_cancer_data = readtable('wbco.csv');
cd ..
X = table2array(breast_cancer_data(:,1:9));
Y = table2array(breast_cancer_data(:,10));
rng(4797);
train_test_partition = cvpartition(Y,'Holdout',0.2,'Stratify',true);
train_idx = training(train_test_partition);
test_idx = test(train_test_partition);
X_train = X(train_idx,:);
X_test = X(test_idx,:);
Y_train = Y(train_idx,:);
Y_test = Y(test_idx,:);

nan_inf_rows = any(isnan(X_train) | isinf(X_train), 2);
X_train = X_train(~nan_inf_rows, :);
Y_train = Y_train(~nan_inf_rows, :);
```

Training initial Takagi-Sugeno model

```
opt = genfisOptions('FCMClustering','FISType','sugeno');
opt.NumClusters = 5;
ts_model = genfis(X_train,Y_train,opt);
```

```
Iteration count = 1, obj. fcn = 10300.7
Iteration count = 2, obj. fcn = 7592.33
Iteration count = 3, obj. fcn = 7020.9
Iteration count = 4, obj. fcn = 5755.96
Iteration count = 5, obj. fcn = 4911.44
Iteration count = 6, obj. fcn = 4689.57
Iteration count = 7, obj. fcn = 4654.69
Iteration count = 8, obj. fcn = 4634.25
Iteration count = 9, obj. fcn = 4609.51
Iteration count = 10, obj. fcn = 4591.23
Iteration count = 11, obj. fcn = 4584.58
Iteration count = 12, obj. fcn = 4583.15
Iteration count = 13, obj. fcn = 4582.87
Iteration count = 14, obj. fcn = 4582.78
Iteration count = 15, obj. fcn = 4582.72
```

Checking initial performance on test set

```
Y_pred_initial = evalfis(ts_model, X_test);
Y_pred_initial(Y_pred_initial>=0.5) = 1;
Y_pred_initial(Y_pred_initial<0.5) = 0;
class_report_initial = classperf(Y_test, Y_pred_initial);
fprintf('Initial Accuracy: %4.3f \n', class_report_initial.CorrectRate);
```

```
Initial Accuracy: 0.971
```

```
fprintf('Initial Sensitivity: %4.3f \n', class_report_initial.Sensitivity);
```

```
Initial Sensitivity: 0.967
```

```
fprintf('Initial Specificity: %4.3f \n', class_report_initial.Specificity);
```

```
Initial Specificity: 0.917
```

Tunning initial model using ANFIS

```
[in,out,rule] = getTunableSettings(ts_model);
anfis_model = tunefis(ts_model,[in;out],X_train,Y_train,tunefisOptions("Method","anfis"));
```

```
ANFIS info:
        Number of nodes: 112
        Number of linear parameters: 50
        Number of nonlinear parameters: 90
        Total number of parameters: 140
        Number of training data pairs: 547
        Number of checking data pairs: 0
        Number of fuzzy rules: 5


Start training ANFIS ...

1        0.138668
2        0.133687
3        0.135736
4        0.133005
5        0.134729
Step size decreases to 0.009000 after epoch 6.
6        0.132625
7        0.134078
8        0.132256
9        0.131669
10       0.131087
```

Checking ANFIS tuned model performance

```
Y_pred_final = evalfis(anfis_model, X_test);
Y_pred_final(Y_pred_final>=0.5) = 1;
Y_pred_final(Y_pred_final<0.5) = 0;
class_report_final = classperf(Y_test, Y_pred_final);
fprintf('Final Accuracy: %4.3f \n', class_report_final.CorrectRate);
```

```
Final Accuracy: 0.978
```

```
fprintf('Final Sensitivity: %4.3f \n', class_report_final.Sensitivity);
```

```
Final Sensitivity: 0.956
```

```
fprintf('Final Specificity: %4.3f \n', class_report_final.Specificity);
```

```
Final Specificity: 0.958
```

Hair Dryer Dataset

Loading dataset and Creating train-test sets

```
clear all
cd DATA
```

```
auto_mpg_data = readtable('hairdryer.csv');
cd ..
X = table2array(auto_mpg_data(:,1:1));
Y = table2array(auto_mpg_data(:,2));
rng(4797);
[train_idx, ~, test_idx] = dividerand(size(X,1), 0.8, 0,0.2);
X_train = X(train_idx,:);
X_test = X(test_idx,:);
Y_train = Y(train_idx,:);
Y_test = Y(test_idx,:);
```

Training initial Takagi-Sugeno model

```
opt = genfisOptions('FCMClustering','FISType','sugeno');
opt.NumClusters = 5;
ts_model = genfis(X_train,Y_train,opt);
```

```
Iteration count = 1, obj. fcn = 617.517
Iteration count = 2, obj. fcn = 463.34
Iteration count = 3, obj. fcn = 409.286
Iteration count = 4, obj. fcn = 271.831
Iteration count = 5, obj. fcn = 228.562
Iteration count = 6, obj. fcn = 179.601
Iteration count = 7, obj. fcn = 130.927
Iteration count = 8, obj. fcn = 95.0993
Iteration count = 9, obj. fcn = 84.6956
Iteration count = 10, obj. fcn = 77.9865
Iteration count = 11, obj. fcn = 74.0098
Iteration count = 12, obj. fcn = 72.8569
Iteration count = 13, obj. fcn = 72.5036
Iteration count = 14, obj. fcn = 72.353
Iteration count = 15, obj. fcn = 72.2744
Iteration count = 16, obj. fcn = 72.2302
Iteration count = 17, obj. fcn = 72.2049
Iteration count = 18, obj. fcn = 72.1903
Iteration count = 19, obj. fcn = 72.1819
Iteration count = 20, obj. fcn = 72.177
Iteration count = 21, obj. fcn = 72.1742
Iteration count = 22, obj. fcn = 72.1726
Iteration count = 23, obj. fcn = 72.1717
Iteration count = 24, obj. fcn = 72.1711
Iteration count = 25, obj. fcn = 72.1708
Iteration count = 26, obj. fcn = 72.1707
Iteration count = 27, obj. fcn = 72.1706
Iteration count = 28, obj. fcn = 72.1705
Iteration count = 29, obj. fcn = 72.1705
```

Checking initial performance on test set

```
Y_pred_initial = evalfis(ts_model, X_test);
rmse_initial = rmse(Y_pred_initial, Y_test);
fprintf('Initial RMSE: %4.3f \n', rmse_initial);
```

```
Initial RMSE: 0.781
```

Tunning initial model using ANFIS

```
[in,out,rule] = getTunableSettings(ts_model);
anfis_model = tunefis(ts_model,[in;out],X_train,Y_train,tunefisOptions("Method","anfis"));
```

```
ANFIS info:
        Number of nodes: 24
        Number of linear parameters: 10
        Number of nonlinear parameters: 10
        Total number of parameters: 20
        Number of training data pairs: 800
        Number of checking data pairs: 0
        Number of fuzzy rules: 5


Start training ANFIS ...

1        0.830271
2        0.830271
3        0.830271
4        0.830271
5        0.830271
6        0.830271
7        0.830271
8        0.830271
9        0.830271
10       0.830271
```

Checking ANFIS tuned model performance

```
Y_pred_final = evalfis(anfis_model, X_test);
rmse_final = rmse(Y_pred_final, Y_test);
fprintf('Final RMSE: %4.3f \n', rmse_final);
```

```
Final RMSE: 0.781
```

Intelligent Systems - Assignment 2 - Neural Network

Imports

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error,
mean_absolute_percentage_error, explained_variance_score
from sklearn.metrics import accuracy_score, recall_score,
precision_score, f1_score, cohen_kappa_score
```

Wisconsin Breast Cancer Original Dataset

Loading dataset and Creating train-test sets

```
data = pd.read_csv('wbco.csv')
data.replace('?', None, inplace = True)
data.dropna(inplace = True)
var_names = ["ClumpThickness", "UniformityOfCellSize",
"UniformityOfCellShape", "MarginalAdhesion",
"SingleEpithelialCellSize", "BlandChromatin", "NormalNucleoli",
"Mitoses"]
data = data.to_numpy()
scaler = MinMaxScaler()
data = scaler.fit_transform(data)
X = data[:,:9]
y = data[:,9]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.2, random_state = 42)
```

Train model

```
regr = MLPClassifier(hidden_layer_sizes = (31,31,31), random_state =
42, max_iter = 500)
regr.fit(X_train, y_train)
```

```
c:\Users\Francisco\Desktop\Uni\2º Ano\SI\Ambiente\Lib\site-packages\
sklearn\neural_network\_multilayer_perceptron.py:690:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500)
```

```
reached and the optimization hasn't converged yet.
  warnings.warn(

MLPClassifier(hidden_layer_sizes=(31, 31, 31), max_iter=500,
random_state=42)
```

Get model predictions

```
y_pred = regr.predict(X_test)
```

Compute classification metrics

```
acc_score = accuracy_score(y_test, y_pred)
print("Accuracy: {:.3f}".format(acc_score))
rec_score = recall_score(y_test, y_pred)
print("Recall: {:.3f}".format(rec_score))
prec_score = precision_score(y_test, y_pred)
print("Precision Score: {:.3f}".format(prec_score))
F1_score = f1_score(y_test, y_pred)
print("F1-Score: {:.3f}".format(F1_score))
kappa = cohen_kappa_score(y_test, y_pred)
print("Kappa Score: {:.3f}".format(kappa))

Accuracy: 0.971
Recall: 0.960
Precision Score: 0.960
F1-Score: 0.960
Kappa Score: 0.937
```

Hair Dryer Dataset

Loading dataset and Creating train-test sets

```
data1 = pd.read_csv('hairdryer.csv', header = None)
data1.columns = ["Voltage","Air_Temperature"]
var_names1 = []
var_names1.append(data1.columns[0])
data1 = data1.to_numpy()
scaler1 = MinMaxScaler()
data1 = scaler1.fit_transform(data1)
X1 = data1[:,0:1]
y1 = data1[:,1]
X_train1, X_test1, y_train1, y_test1 = train_test_split(X1, y1,
test_size = 0.2, random_state = 42)
```

Train model

```
regr1 = MLPRegressor(hidden_layer_sizes = (20,20), random_state = 42,
max_iter = 1000, learning_rate_init = 0.001)
regr1.fit(X_train1, y_train1)
```

```
MLPRegressor(hidden_layer_sizes=(20, 20), max_iter=1000,
random_state=42)
```

Get model predictions

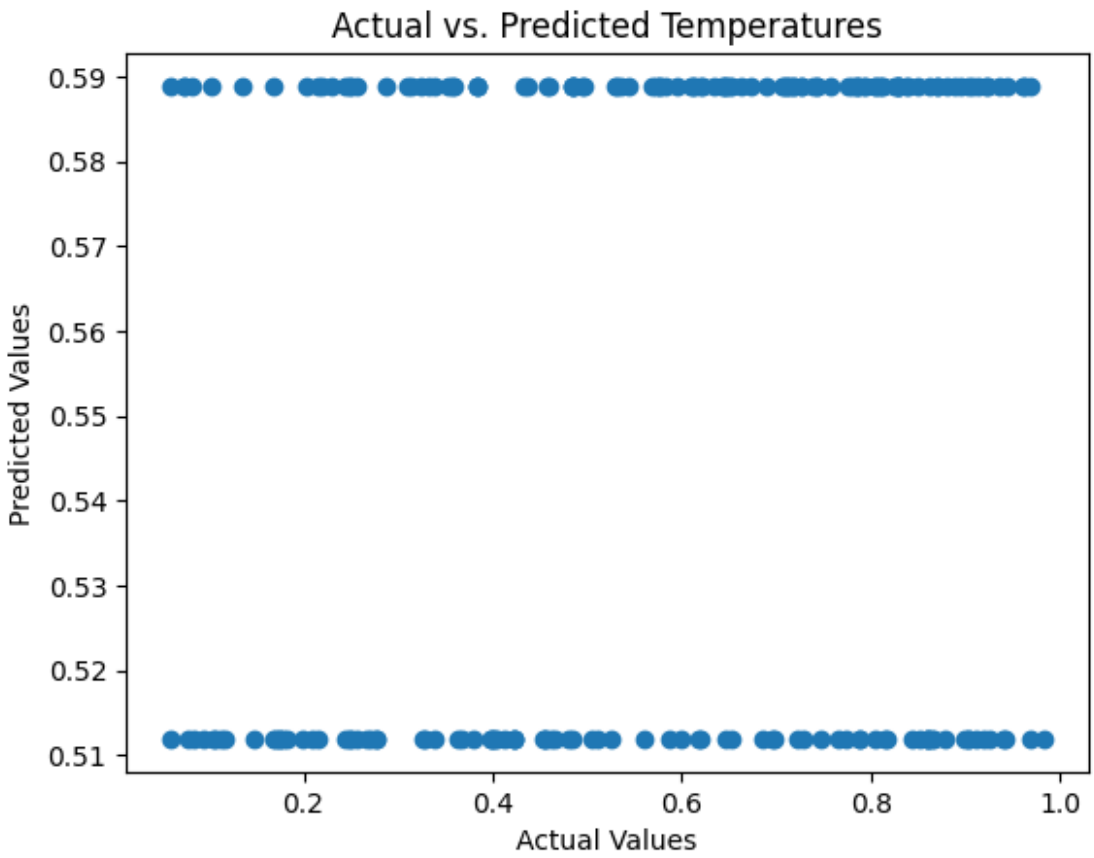
```
y_pred1 = regr1.predict(X_test1)
```

Compute regression metrics

```
mse1 = mean_squared_error(y_test1, y_pred1)
print("Mean Squared Error: {:.3f}".format(mse1))
mape1 = mean_absolute_percentage_error(y_test1, y_pred1)
print("Mean Absolute Percentage Error: {:.1f}%".format(mape1*100))
exp_var1 = explained_variance_score(y_test1, y_pred1)
print("Explained Variance Score: {:.3f}".format(exp_var1))
```

```
Mean Squared Error: 0.067
Mean Absolute Percentage Error: 80.5%
Explained Variance Score: 0.029
```

```
plt.scatter(y_test1, y_pred1)
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title("Actual vs. Predicted Temperatures")
plt.show()
```



GITHUB

https://github.com/FranciscoCarvalho26/SI