MECE 694: Applied Computational Intelligence for Engineers
Assignment 4
Due Date: Dec. 8 at 5 pm

Instructor: Milad Nazarahari (nazaraha@ualberta.ca)
TA: Armin Bonakdar (bonakdar@ualberta.ca)

1. Guidelines:

- Submit the finalized codes (.mat file) as well as your report (.pdf file), including results and engineering analysis, as a zip file on eClass.
- Include the assignment rubric on the first page of your report.
- You don't need to include the finalized codes in your report. Only include the qualitative answers or parameter investigation results.
- Your report must be produced with "Overleaf" using this template.
- Codes must also be uploaded to your personal GitHub page under "MECE 694, Assignment #." Your repository must be private, with the course instructor and TA added as a collaborator.
- **Remark:** Failing to comply with the above requirements will result in an assignment mark reduction.

Assignment 4 has three sections:

1. Fuzzy Inference for Temperature Control.
2. Using the ANFIS tuning method to design a fuzzy system for time-series prediction.
3. Using the ANFIS tuning method to design a fuzzy system for noise cancelling (Optional).

For some sections, sample codes will be provided for you, and you need to modify the codes to obtain the required results.

2. Fuzzy Inference for Temperature Control

Design and implement a fuzzy inference system in MATLAB's Fuzzy Logic Toolbox to control the temperature of a heating system based on input variables. Analyze the performance of the fuzzy system and provide insights into its effectiveness.

Imagine you are in charge of a heating system in a building, and your goal is to maintain a comfortable temperature in a room. You have two input variables:

- Temperature Error (TE): The difference between the desired temperature and the current temperature in degrees Celsius. Typical range: -10°C to 10°C (we live in Edmonton!!!).
- Rate of Temperature Change (RTC): The rate at which the temperature changes in degrees Celsius per minute. Typical range: -5°C/min to 5°C/min.

You also have one output variable:

- Heater Control (HC): The degree to which the heater should be turned on, represented as a percentage (0% for off, 100% for full power).

Question 1:

- Impalement the following in MATLAB's Fuzzy Logic Toolbox.
  - Fuzzy Set Definitions: Define fuzzy sets for the input variables (TE and RTC) and the output variable (HC) with appropriate membership function shapes and ranges (# of membership functions and their shapes/parameters are your choice, but should be defendable).
  - Fuzzy Rules: Create a set of fuzzy rules to map input variables to the output variable.
  - Fuzzy Inference: Implement the fuzzy inference system using MATLAB's Fuzzy Logic Toolbox. Use the Mamdani method for inference.
  - Defuzzification: Apply defuzzification to determine the crisp output value (HC) from the fuzzy output.
- Demonstrate the architecture of your fuzzy system (you can attach screenshots of your fuzzy system in your report) and describe your rationale for the selected architecture and parameters.
- Simulate the fuzzy system for various scenarios by specifying TE and RTC values. Observe how the heater control responds to different temperature conditions. Does your fuzzy system work as expected?

3. Time Series Prediction

For some dynamic systems, their output at each time instant t can be characterized only based on their previous outputs. In mathematical terms:

$$x_t = \mathrm{g}(\{x_\tau | \tau_1 \le \tau \le \tau_2 \le t\}) \, , \; t \ge 0$$

Now assume that we work in discrete time; therefore, instead of using all-time instances between τ_1 and τ_2 to predict x_t, we only use system output in the past in a number of specific time instances:

$$x_t = \mathrm{g}\big(\{x_\tau | \tau \in \{t-\delta_1, t-\delta_2, \; \cdots, t-\delta_n\}\}\big), \; t \ge 0$$

where δ_1, δ_2, …, δ_n are the delay values.

The **GOAL** in time series prediction is to find the function **g** using a predetermined number of delays δ_1, δ_2, …, δ_n such that we can predict the current dynamics of the system x_t using its previous dynamics.

For example, consider the following mathematical model:

$$x_t = \mathrm{g}(x_{t-3}, x_{t-6}), \; t \ge 0$$

which means we can predict the output at each time instant t using samples $t-3$ and $t-6$ from system output history. Also, we know that the first time instant that we can make a prediction is $t = 6$ (if we want to make a prediction at $t = 5$, we need to have x_{-1}, which is not meaningful).

Remark: in all previous topics related to nonlinear function approximation, the mathematical model that we were trying to build was:

$$y_t = g(x_t),\ t \geq 0$$

therefore, time series prediction for dynamic systems, in general, is a nonlinear function approximation problem where past outputs of the system can also be considered as the input features.

We can solve the nonlinear function approximation problem using a variety of tools. One solution is the TSK fuzzy inference system tuned by the ANFIS method or other search strategies such as Genetic Algorithm. Yet before that, we need to create input and output data.

For instance, to create input and output data for $x_t = g(x_{t-3}, x_{t-6}),\ t \geq 0$, we can perform the following:

Time Instant	Input Matrix		Output Matrix
	x_{t-3}	x_{t-6}	x_t
6	x_3	x_0	x_6
7	x_4	x_1	x_7
…	…	…	…
T	x_{T-3}	x_{T-6}	x_T

Where the number of rows is T – 6 + 1.

Question 2:

- (code) Given a time series $x = (x_1, x_2, \cdots, x_T)$, create a function named "TimeSeries_Data" which accepts time series x and a vector containing a number of delays δ_1, δ_2, …, δ_n and returns proper input and output for a time series prediction problem similar to the above example. Your function must work for any delay vector.
  - For this example, you will be given the time-series x which contains the data obtained from Mackey-Glass equation (learn more about Mackey-Glass equation here).
  - You can use delay vector [6, 12, 18, 24] for answering Questions 3 and 4. But your code must be usable for any other delay vector too.

- For **Questions 2** and **3**: create a script name "TimeSeries_ANFIS" and add your codes to the script.

Question 3:

- (code) After creating input and output, divide data into training, validation, and testing sets randomly (you can use MATLAB built-in functions for this purpose).

- Decide about the percentage of the data you want to use for training, validation, and testing.

Question 4:

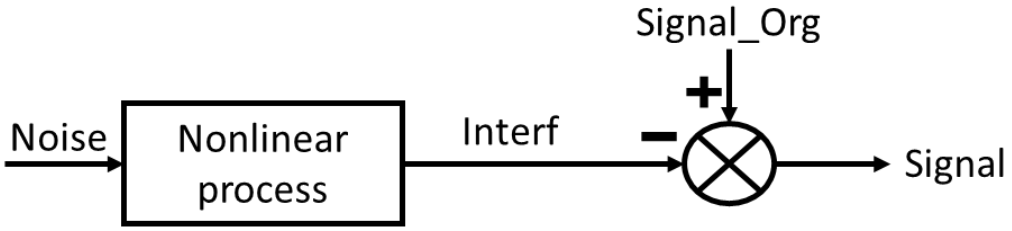
- (code and report) Use the sample code provided for TSK FIS tuning using ANFIS to build a FIS to perform time series prediction.
  - Use training and validation data for training, and tune the FIS and then investigate the FIS behaviour for test data.
  - Select proper ANFIS parameters and report them.
  - Create the raw FIS using "grid partitioning", "subtractive clustering", and "fuzzy c-means". For each method, try at least three different parameter values by changing at least one parameter at a time and report the root-mean-square of the error between predicted and true values of the time series for training and testing datasets in a table.
  - Plot the predicted output with FIS for training and testing data against the true values of the time series for the best parameter settings obtained by each method.
  - (If we haven't covered RBF in the lectures, ignore this question) Compare the time series prediction between TSK FIS tuned by ANFIS and Radial Basis Function (discussed in earlier weeks). Select the proper settings for RBF.
  - Investigate the behaviour of the best FIS for two other sets of delay values.

4. Noise Cancellation Using ANFIS (Optional – Will not be Marked)

In this section, you will learn how to use an optimized FIS to remove the noise which has corrupted an audio signal.

The provided "Noise_Cancelling_Incomplete1" contains the codes for loading an audio file, playing the file, corrupting it with white noise, playing the noisy file, and then instructions for using ANFIS to remove the noise.

Assume that the original clean audio signal, called "Signal_Org", cannot be measured without an interference signal, called "Interf", which is generated from a noise source, called "Noise", via a certain unknown nonlinear process.



Assume that the interference signal "Interf" that appears in the measured signal "Signal" is generated via an unknown nonlinear equation; for instance:

$$Interf(k) = (2*\sin(2*Noise(k)) * Noise(k-1)) / (1+Noise(k-1)^2) + 0.2 * \cos(Noise(k)) / \exp(\sin(2*Noise(k-1)))$$

The measured signal, "Signal", is the sum of the original information signal, "Signal_Org", and the interference, "Interf". However, we do not know "Interf". The only signals available to us are the noise signal "Noise" and the measured signal "Signal".

In Section 1 of the code "Noise_Cancelling_Incomplete1", you can play the original audio file, "Signal_Org", and the noisy version, "Signal" which is equal to "Signal_Org" + "Interf", to see the effect of adding noise to the data.

Question 5:

- (code) Use ANFIS to identify the nonlinear relationship between "Interf" and "Noise". As "Interf" is not directly available, we will assume that "Signal" is a corrupted version of "Interf" to perform FIS tuning using ANFIS.
- The input and output for training with ANFIS are created in Section 2 of "Noise_Cancelling_Incomplete1".
- Implement ANFIS with your choice of settings and your choice of method for generating raw FIS.
- (Bonus Question) Can you find a quantitative metric to measure the quality of the denoising? If yes, use your metric to investigate the effect of ANFIS and method for generating raw FIS parameters on denoising.