Intelligent Systems Engineering
ECE 449
Fall 2023

Laboratory Exercise #4

Resources

ECE 449 Slides from last year (Dr. Musilek)

- Introduction to Fuzzy Sets
- Membership Functions
- Rue-Based Systems

R. Kruse et al., *Computational Intelligence: A Methodological Introduction, 3rd Ed.* (text)

- Chapter 15, Fuzzy Logic (up to15.4.2)
- Chapter 19, Approximate Reasoning

Intro to Fuzzy Logic Slides (Dr. Dick)

skfuzzy Documentation

- Examples https://pythonhosted.org/scikit-fuzzy/auto_examples/index.html

Summary

Using the skfuzzy library in Python, build a fuzzy tree to recommend a tip amount at a restaurant. This extends the "Tipper" example in the skfuzzy documentation, which is just a single fuzzy rulebase, that accepts two inputs ("Food Quality" and "Service Quality"). However, the source of the Food and Service Quality inputs are never specified. You will build a fuzzy tree whose earlier stages compute those inputs, which then feed to the "Tipper" rulebase.

Discussion

A fuzzy rule-based system (also commonly called a Fuzzy Inferential System, FIS) differs from the AIs we have studied up until now by being *declarative*. The FIS designer actually specifies the fuzzy membership functions for inputs and outputs, and the rules that relate them together; nothing is learned from data (at least for this project). The "Tipper" FIS on the skfuzzy Examples page is a good example. Two real-valued inputs, and a real-valued output are chosen, and three fuzzy sets for each one are created. Then some rules are designed to relate inputs to outputs, in a simple common-sense relationship.

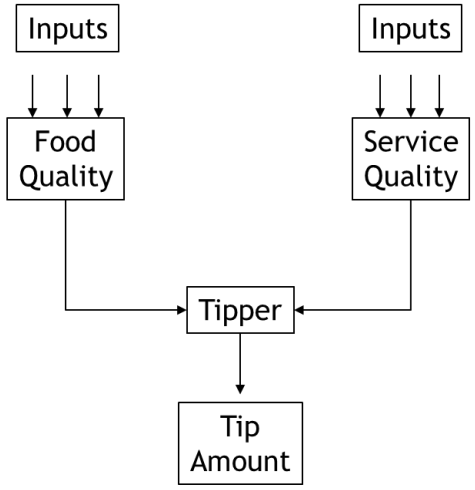
However, what exactly is "Food Quality" or "Service Quality?" In our discussions around what "quality" even means, we've touched on the fact that it's not directly measurable, so where are the inputs coming from? The easy answer is, it's just a subjective measure; how do you *feel* about each one? The Tipper FIS will then recommend a tip based on your feelings.

A better answer is that Food Quality and Service Quality can be multi-dimensional vectors. Let's say that Food Quality breaks down into *food temperature*, *food flavor*, and *portion size* (all theoretically measurable). Likewise, let's say Service Quality breaks down into *attentiveness*,

friendliness, and *speed of service*. You could measure attentiveness by the length of time between table visits; friendliness is subjective (maybe you could assess body language and vocal tone); and speed of service comes down to elapsed time between order and delivery. The key point, though, is that each one is much more concrete and understandable than a nebulous statement of "quality." Okay, let's define our FIS in terms of these six inputs.

Or not, as we now hit a classic problem in Big Data: the *curse of dimensionality*. For FIS specifically, this means that the number of rules needed to account for all the possible input combinations grows exponentially with the number of inputs. Technically, you have (number of fuzzy sets per dimension) ^ (number of dimensions), or $3^6 = 729$ rules, if we assume 3 fuzzy sets per input. This suddenly makes our simple FIS much less simple. So now what?

One of the possible solutions is to break the big FIS down into smaller components. There are several theoretical frameworks for doing this; the one I would like you to explore is a *fuzzy tree*, a structure where multiple 1st-stage FISs feed into one 2nd-stage FIS, as in the figure below. In this figure, we are using an FIS to combine the Food Quality attributes into one scalar measure, and the same is happening to the Service Quality attributes. As you probably realized, these are now producing the Food Quality and Service Quality inputs for the Tipper FIS, and so we feed them right into that FIS.



To accomplish this in skfuzzy, you will have to create a data structure in Python that will hold the `control` objects for each FIS, and pass the outputs of the Food Quality and Service Quality FISs as inputs to Tipper during a `controlSimulation`.

As an aside, this style of fuzzy tree is also the recommended architecture for building your intelligent agent in the XFC game during your group project. Dr. Dick will be preparing an example game agent using this approach, to be released in early November.

Requirements

Create a fuzzy tree using skfuzzy to implement the tip recommender as discussed, using the six input attributes above. The user should be able to input values for the six measures, and then receive a recommended tip (as a percentage of the final bill). The user should then be prompted

to enter another set of measures (in theory representing a different meal at perhaps a different restaurant). The program should terminate when the user does not wish to enter any more values. Remember to validate your inputs – the user should not enter out-of-range inputs, or anything but legal values for each attribute. As usual, this should be performed in an automated pipeline.

You will need to decide on the rules for each FIS, and the number of and parameters for fuzzy sets in each of the six input dimensions, the 1st-stage FIS outputs, and the final Tipper output. I recommend using center-of-gravity defuzzification in all three FIS (yes you should defuzzify the Food Quality and Service Quality outputs; this makes it possible to follow the Tipper design from the skfuzzy examples).

Submission

All of the above should be automatically performed in a Python-based pipeline, and submitted as a Jupyter notebook. It is NOT necessary that the pipeline() object from scikit-learn be used in this assignment, although it is not prohibited. Submit the notebooks via eClass in your lab section before the deadline.

Grading

Requirements satisfied via fuzzy tree:	50%
Pipeline correctly executes:	50%