Coursework Indicative Fuzzy Inference System

Dr. Archie Singh **Khuman**

INSTITUTE OF ARTIFICIAL INTELLIGENCE · GAMES, MATHS & ARTIFICIAL INTELLIGENCE · GH6.57

De Montfort University, Gateway House, Leicester, United Kingdom, LE1 9BH

+44 (0)116 250 6251 | arjab.khuman | arjab singh khuman | archie khuman

"Be inspired or be inspirational - Either way, great things can be accomplished."

Outline for the Presentation...

1. Getting Started

2. Fuzzy Inference System

MATLAB - Getting Started

- We will be creating a new *fuzzy inference system* (**fis**).
- The system will be *indicative* of the coursework requirements.
- The coursework will be released around **Week 4**.
- Be sure to watch the accompanying coursework *presentation recording* when made available.
- This will provide a full *overview* of what to expect and how best to start.
- *Skeleton code* will be made available and also past *reports* of varying quality.
- Which you can use for **inspiration** & **guidance**.

MATLAB - Getting Started

- The system we will be creating will be *far* from perfect.
- It will leave plenty of room for *improvement* & *enhancement*.
- It is **NOT** the system you create that gets the marks, it's **HOW** you go about creating the system that gets the marks.
- The report you will be writing will need to describe the *journey* of the system, the **development life cycle**.
- The coursework requirements are that the *minimal* amount of fuzzy inputs be **3** and the *minimal* outputs be at least **1**.
- The *report* is what really carries the marks, so take *care* and your *time* when writing it.

MATLAB - Getting Started

- If you did install **MATLAB** on your own devices, make sure that you also included the **Fuzzy Logic Toolbox** as part of the install.
- The **Fuzzy Logic Toolbox** contains the *functions calls* that we will be making use of.
- Without the toolbox, your commands will not be executed, and your scripts will **NOT** compile & run.
- **Make** sure to also watch the *video tutorial*.
- The tutorial will offer an additional *understanding*, and you can follow along in your own time.

Outline for the Presentation...

Fuzzy Inference System

- For this lab we will begin implementation of a **rule-base** & **inference engine**.
- Your coursework will involve you creating *your own* fuzzy inference systems, they are all started in much the same way.
- You will get to decide on *what* your system will be doing.
- Use your creativity and choose something that genuinely *interests* you.
- Many previous students have gone on to implement fuzzy systems into their *development projects*.
- A fuzzy system is an **AI** paradigm, so *more* marks if it's applicable to your project.

Fuzzy Inference System

- Open a **new script file**, the code for this session can also be found in the lab folder.
- We will also be making use of an *external* excel file to hold our system input data.
- This too will also be available in the **lab folder**.
- **Make** sure to **watch** the accompanying *video tutorial*.
- By the end of the lab, you will have a *fully functioning* **fis**.
- You could use this code as the *basis* of your coursework if you wish.
- The system will **satisfy** the *minimum* requirements for the coursework brief.

System Overview

- We will be creating a **Heating Control Unit**.
- There are **3** inputs with **1** output:
- **Input 1 - Day of Year**, has **5** fuzzy associated to it:
  Winter(1), *Spring*, *Summer*, *Autumn* & *Winter(2)*
- **Input 2 - Time of the Day**, has **5** fuzzy associated to it:
  Twilight(1), *Morning*, *Afternoon*, *Evening* & *Twilight(2)*
- **Input 3 - Temperature**, has **5** fuzzy associated to it:
  Very Cold, *Cold*, *Moderate*, *Warm* & *Very Warm*
- **Output 1 - Heating**, has **4** fuzzy sets associated to it:
  Off, *Low*, *Moderate* & *High*

Declaration of System

- The code from this session can be found in the **lab folder**.
- In your new **script file**, at the top, *place* the following command:

```
a = newfis('Control Unit');
```

- Once the system has been created, we will be revisiting the `newfis` command and *modifying* it.
- To change the *configuration* settings to see how this can impact on the performance of the system.
- You will now need to *declare* **Input 1** and *populate* it with membership functions:

Input 1 - Day of Year

```
a=addvar(a,'input','Day of Year',[0 365]);

a = addmf(a, 'input', 1, 'Winter(1)', 'gaussmf', ...
[25 0]);

a = addmf(a, 'input', 1, 'Spring', 'gaussmf', [25 ...
105.5]);

a = addmf(a, 'input', 1, 'Summer', 'gaussmf', [25 ...
197.5]);

a = addmf(a, 'input', 1, 'Autumn', 'gaussmf', [25 ...
289]);

a = addmf(a, 'input', 1, 'Winter(2)', 'gaussmf', ...
[25 365]);
```

Input 1 - Day of Year

- How many *seasons* are there in a year?
- Why are there **5** fuzzy sets? Why are there **2** *Winter* fuzzy sets?
- The input itself is a *continuous* input.
- When we reach the *end* of a calendar year, we **DO NOT** continue on for infinity.
- We *restart* from the **beginning** of the year at January 1^{st} and then continue again to the **end** of the year.

Input 1 - Day of Year

- Imagine the **right-most** end of the plot *wrapping* around to the **left-most** end.
- You have effectively created a *continuous* representation.
- *Winter(1)* & *Winter(2)* although two *separately* coded fuzzy sets are actually the **same** season when we wrap them around to one another.
- That is why we need **5** fuzzy sets to replicate **4** seasons.
- The *distribution* of the fuzzy sets was chosen purely using an **arbitrary** assumption.
- Always begin with some *initial* values for settings, after which you can *fine-tune* through testing.

Input 2 - Time of Day

```
a = addvar(a, 'input', 'Time of Day', [0 1440]);

a = addmf(a, 'input', 2, 'Twilight(1)', 'trapmf', ...
[0 0 60 240]);

a = addmf(a, 'input', 2, 'Morning', 'trapmf', ...
[120 300 660 780]);

a = addmf(a, 'input', 2, 'Afternoon', 'trapmf', ...
[660 780 1020 1140]);

a = addmf(a, 'input', 2, 'Evening', 'trapmf', ...
[1020 1140 1320 1440]);

a = addmf(a, 'input', 2, 'Twilight(2)', 'trapmf', ...
[1200 1380 1440 1440]);
```

Input 2 - Time of Day

- For **Input 2** I have decided upon **5** fuzzy sets.
- This is another *arbitrary* decision that I made to begin with.
- I *decided* that I would use the number of minutes in a day rather than the hours.
- This would be something I would then have to *justify* and provide a *rationale* for in the report.
- There are **1440** minutes in a 24 hour day, I broke this down to **5** *segments*.
- In much the same way as the previous input, **Input 2** is also *continuous*.

Input 2 - Time of Day

- *Twilight(1)* & *Twilight(2)* are effectively the **same** segment, just on either ends of the input.
- The choice of membership functions being trapezoidal, *arbitrary*.
- The distribution of the fuzzy sets, *arbitrary*.
- Everything is arbitrary to begin with as this is the **first iteration** of the system.
- After inspecting the *output performance* one can then make attempts to configure and tweak aspects of the system.

Input 3 - Temperature

```
a = addvar(a, 'input', 'Temp', [-20 40]);

a = addmf(a, 'input', 3, 'Very Cold', 'trapmf', ...
[-20, -20, -1, 0]);

a = addmf(a, 'input', 3, 'Cold', 'trapmf', [-2, ...
0, 4, 6]);

a = addmf(a, 'input', 3, 'Moderate', 'trapmf', ...
[4, 6, 10, 12]);

a = addmf(a, 'input', 3, 'Warm', 'trapmf', [10, ...
14, 18, 22]);

a = addmf(a, 'input', 3, 'Very Warm', 'trapmf', ...
[18, 22, 40, 40]);
```

Input 3 - Temperature

- **Input 3** is **NOT** continuous because temperature is not continuous; that's *common sense* and **factual**.
- Why does it start at **minus 20** and finish at **40**?
- Because **I** decided it could, it made sense to me to provide *extreme* values.
- Again, I would have to justify and provide my *reasoning* for this in the report.
- This is my system, the report needs to reflect my *thought processes*.
- The *why* I did what I did & the *how* I went about it.

Output - Heating

```
a=addvar(a,'output','Heating (%)',[-5 100]);

a = addmf(a, 'output', 1, 'Off', 'trapmf', [-5 -5 ...
0 0]);

a = addmf(a,'output',1,'Low','trimf',[0 15 33]);

a = addmf(a,'output',1,'Moderate','trimf',[33 ...
49.5 66]);

a = addmf(a,'output',1,'High','trapmf',[66 83 100 ...
100]);
```

Output - Heating

- The **Output** is with regards to how much the heating will be *on*.
- You'll notice that I have *implemented* an **Off** state.
- There may be instances where the heater will be completely off, like in *Summer* for example.
- Notice that there is no overlap in my fuzzy sets for the output.
- *Arbitrary*, after testing I can then go back and *reconfigure* and tweak.
- I need an initial configuration to begin with, after which changes can be made.

Rules

- The rule-base is small and far from finished, but a *start*.
- You will need to add to it improve the system's performance and ability.

```
rule1 = [1 1 2 3 1 1]; rule2 = [1 1 3 3 1 1];

rule3 = [5 5 2 3 1 1]; rule4 = [5 5 3 3 1 1];

rule5 = [5 1 2 3 1 1]; rule6 = [5 1 3 3 1 1];

rule7 = [1 5 2 3 1 1]; rule8 = [1 5 3 3 1 1];

rule9 = [0 0 1 4 1 1]; rule10 = [0 0 5 1 1 1];
```

- Did you *notice* `rule9` & `rule10`?

Rules

- Your *fuzzy rules* do not need to make use of **ALL** the inputs.
- Notice the **0** in position of **Input 1** and **Input 2**, but values for **Input 3** and the **Output**.
- Input 3 is the *temperature*, and this system is with regards to *controlling* the heating.
- It stands to reason that I **DO NOT** need to be *concerned* with the *time of year* or *time of day*, if the temperature is *very warm or very cold*.
- `rule9` & `rule10` allow for the heating to be either completely **OFF** or completely **ON** to a *high* amount.

Rules

```
rule11 = [0 2 1 4 1 1]; rule12 = [0 2 2 4 1 1];
rule13 = [0 2 4 2 1 1]; rule14 = [0 3 1 4 1 1];
rule15 = [0 3 2 4 1 1]; rule16 = [0 3 3 3 1 1];
rule17 = [0 3 4 2 1 1]; rule18 = [0 3 5 1 1 1];
```

- What about rules `rule11` through to `rule18`?
- Same *thought process* & *reasoning*, apply your own **subjectivity**.
- Create rules that make sense to **YOU**.

Data

```
ruleList = [rule1; rule2; rule3; rule4, rule5; ...
rule6; rule7; rule8; rule9; rule10; rule11; ...
rule12; rule13; rule14, rule15; rule16; rule17; ...
rule18;];

a = addrule(a, ruleList);

showrule(a)

data = ('ControlUnitData.xlsx');

testData = xlsread(data);
```

Data

- We collect the individual rules in a *matrix* called `ruleList`
- We then *pass* matrix to the `addrule` function to be able to *add* it to our system.
- When you create any *additional* rules, they must be placed into the `ruleList` matrix.
- You might have noticed `'ControlUnitData.xlsx'` being *passed* to a variable called `data`.
- `'ControlUnitData.xlsx'` is an **external excel** file that contains the initial input values for the system.

Data

- Open this excel file *outside* of MATLAB.
- You can see **Input 1** in **Column A**, **Input 2** in **Column B** and **Input 3** in **Column C**.
- **Column D** is purely for our *reference* and provides an exact date and time; this is **NOT** passed into our system.
- When we declared our inputs we also had to specify the *ranges* - the range of the x-axis.
- Each column of data is *within* the ranges for their respective inputs.
- If I had a value that exceeded the range, MATLAB would output a *warning*.

Data

- You would have also *noticed* `testData = xlsread(data);`
- The `xlsread()` function is needed to *read* in the **excel file**.
- It is at this point in the code that system has *access* to data,
- The same data that we will use to **test** the system.
- Rather than manually declaring input values as we have done previously, it is *advised* that you adopt this method.
- We can have the input data fed into the system, *row-by-row*, and the output displayed in the **Command Window**.
- Equally, we can also have the output of the system *written* to an excel file.

Data

```
for i=1:size(testData,1)

output = evalfis([testData(i, 1), testData(i, 2), ...
testData(i, 3)], a);

fprintf('%d)In(1): %.2f, In(2)%.2f, In(3)%.2f => ...
Out: %.2f nn',i,testData(i, 1),testData(i, ...
2),testData(i, 3), output);

xlswrite('ControlUnitData.xlsx', output, 1, ...
sprintf('F%d',i+1));

end
```

Data

- The **for loop** processes the data and *feeds* it into our system, row-by-row.
- As long as the excel file is in the **same** directory as the **script file**, the system will know where to look.
- We have a variable called `testData` that holds the *excel information*.
- It is this that is passed into the for loop, where indexing starts at **1** and finishing at the *size* of `testData`.
- In the for loop we make use of the `evalfis` command, which will be needed in **ALL** fuzzy systems.
- Each input row is being passed into `evalfis` and *written* to `output`.

Data

- The `fprintf` is the command that *prints* the output of the system to the **Command Window**.
- The `xlswrite` function is writing the same output from **Command Window** to the same *excel file* that we read the input values from.
- Specifically, it is writing them to **Column F** from row **2** onwards down.
- Be sure to **close** the excel when you *run* your system.
- If you have it open MATLAB will throw an **error** message.
- Once compiled, if you were to open the excel file you would *see* the output of the system.

Visuals

```
ruleview(a)

figure(1)

subplot(4,1,1), plotmf(a, 'input', 1)

subplot(4,1,2), plotmf(a, 'input', 2)

subplot(4,1,3), plotmf(a, 'input', 3)

subplot(4,1,4), plotmf(a, 'output', 1)
```

Visuals

- The `ruleview(a)` will allow for you to *inspect* the rule-base and interact with it.
- *Clicking* & *dragging* the red input lines, you can see which rules are *firing* and being *engaged*.
- The `figure(1)` handler creates an *instance* of a figure for which you can populate with *plots*.
- The *subplots* collectively display all **3 Inputs** and the **Output** to the figure handler.
- If I declared another figure handler `figure(2)` **after** `figure(1)`.
- Any new plots would be then placed on `figure(2)` and **NOT** `figure(1)`.

Fuzzy Inference System

- Compile & run the **script file**, what do you see?
- It compiled and ran successfully, albeit with a few warnings.

```
Warning: No rules fired for Output 1. Defuzzified ...
output

value set to its mean range value 47.5.
```

- The reason why is because we **DO NOT** have rules that *engage* with some of our input data.
- The rule-base is too small and ineffective, you need to **add** *more* rules to cater for your input data.
- Equally, you could create *new* input data that would *work* with your rule-base.

Fuzzy Inference System

- Open the `'ControlUnitData.xlsx'` file outside of MATLAB.
- You can see the output of our system has been written to **Column F**.
- Having the ability to specify where to *write* can come in very *handy*.
- Looking at the output values themselves, you can ignore instances of **47.5**.
- They are effectively *dump values* which have not been calculated.
- **Close** the excel file and go back to the **script file**.

Fuzzy Inference System

- `newfis` by default is *configured* to have the following parameter settings:

  `FISType` = Mamdani

  `AndMethod` = min

  `OrMethod` = max

  `ImplicationMethod` = min

  `AggregationMethod` = max

  `DefuzzificationMethod` = centroid

Fuzzy Inference System

- Go back to your: `a = newfis('Control Unit');`
- **Modify** the `newfis` declaration so looks like the following:

```
a = newfis('Control Unit', ...
'DefuzzificationMethod', 'mom');
```

- We have changed the *defuzzification method* from `centroid` to `mom`
- If you are unsure what that the defuzzification methods do, click **HERE**.
- **DO NOT** compile & run just yet.

Fuzzy Inference System

- Go to the *for loop*, where you saw the `xlswrite` function.
- Find the `sprintf('F%d',i+1)` part of the `xlswrite` function.
- **Modify** it so it looks like the following, change the **F** to a **G**:

  ```
  sprintf('G%d',i+1)
  ```

- You have effectively *shifted* the column where the output will be *written* to.
- By doing this, you will be able to *compare* & *contrast* the output values based on different configurations.
- Now compile & run the system, once done **open** the excel file *outside* of MATLAB.

Fuzzy Inference System

- You can now see the output of `centroid` *defuzzification method* in **Column F**.
- And the `mom` defuzzification method in **Column G**.
- Simply changing the *defuzzification method* does change the output values for *some* inputs.
- It is up to you to *justify* which configuration is more aligned with what should be *expected*.
- **Modify** the `newfis` declaration so looks like the following:

```
a = newfis('Control Unit', ...
'DefuzzificationMethod', 'lom');
```

- *Update* the `xlswrite` function from **G** to **H**.

Fuzzy Inference System

- Same as before, once compiled & run, **open** the excel file *outside* of MATLAB.
- Are these values different too? In some cases, **yes**.
- **Modify** the `newfis` declaration so looks like the following:

```
a = newfis('Control Unit', ...
'DefuzzificationMethod', 'som');
```

- Update the `xlswrite` function from **H** to **I**.
- Repeat and *inspect* the values, compare and contrast.

Fuzzy Inference System

- We've only changed the *defuzzification methods*.
- There are many other things to *change*, *tweak*, *modify* & *experiment* with to improve upon your system's performance.
- Using the `ruleview` function, we can see that rules associated to **Input 1** do not really *make use* of many fuzzy sets.
- Consider modifying rules, making use of the **OR** operator, not just exclusively **AND**.
- When you create your own systems, **you** will *define* what you *think* is *appropriate*.
- Never under estimate the quality of *testing* on a system.
- You can **never** test *enough*.

Coursework Indicative Fuzzy Inference System

Dr. Archie Singh **Khuman**

INSTITUTE OF ARTIFICIAL INTELLIGENCE · GAMES, MATHS & ARTIFICIAL INTELLIGENCE · GH6.57

De Montfort University, Gateway House, Leicester, United Kingdom, LE1 9BH

+44 (0)116 250 6251 | arjab.khuman | arjab singh khuman | archie khuman

"Be inspired or be inspirational - Either way, great things can be accomplished."