

TERM: M1-RAIA

SEMESTER: 1

AY: 2022-2023

Abdelbacet Mhamdi

Dr.-Ing. in Electrical Engineering

Senior Lecturer at ISET Bizerte

abdelbacet.mhamdi@bizerte.riset.tn

ARTIFICIAL INTELLIGENCE - PART 1

LAB MANUAL



Higher Institute of Technological Studies of Bizerte

Available @ <https://github.com/a-mhamdi/jlai/>

--- HONOR CODE ---

THE UNIVERSITY OF NORTH CAROLINA AT CHAPEL HILL

Department of Physics and Astronomy

<http://physics.unc.edu/undergraduate-program/labs/general-info/>

“During this course, you will be working with one or more partners with whom you may discuss any points concerning laboratory work. However, you must write your lab report, in your own words.

Lab reports that contain identical language are not acceptable, so do not copy your lab partner’s writing.

If there is a problem with your data, include an explanation in your report. Recognition of a mistake and a well-reasoned explanation is more important than having high-quality data, and will be rewarded accordingly by your instructor. A lab report containing data that is inconsistent with the original data sheet will be considered a violation of the Honor Code.

Falsification of data or plagiarism of a report will result in prosecution of the offender(s) under the University Honor Code.

On your first lab report you must write out the entire honor pledge:




The work presented in this report is my own, and the data was obtained by my lab partner and me during the lab period.

On future reports, you may simply write “Laboratory Honor Pledge” and sign your name.”

Contents

1	<i>Julia</i> Onramp	1
2	Tipping Problem	2
3	Selection Process - Case of RAIA	11
4	Fuzzy Control of an Articulated System	13
5	Exploring the fundamentals of ANN	15
6	Binary Classifier using ANN	16
7	Project Assessment	18

In order to activate the virtual environment and launch **Jupyter Notebook**, we recommend you to proceed as follow

- ① Press simultaneously the keys  &  on the keyboard. This will open the dialog box **Run**;
- ② Then enter `cmd` in the command line and confirm with  key on the keyboard;
- ③ Type the instruction `jlai.bat` in the console prompt line;



- ④ Finally press the  key.

LEAVE THE SYSTEM CONSOLE ACTIVE.

1 | *Julia* Onramp

Student's name

Score /20

Detailed Credits

Anticipation (4 points)
Management (2 points)
Testing (7 points)
Data Logging (3 points)
Interpretation (4 points)

Goals

- ★ Learn the essentials of *Julia* on commonly used features & workflows.



The notebook is available at <https://github.com/a-mhamdi/jlai/> → Codes → *Julia* → Part-1 → *julia-onramp.ipynb*

2 | Tipping Problem

Student's name

Score /20

Detailed Credits

Anticipation (4 points)
Management (2 points)
Testing (7 points)
Data Logging (3 points)
Interpretation (4 points)

Goals

- ★ Construct algorithms to help decide in given ambiguous situation.



The notebook is available at <https://github.com/a-mhamdi/cosnip/> → Matlab → Fuzzy
→ Tipper.*

Matlab is an interesting tool for engineers. All you need in *Matlab* is the basics pieces of information needed to solve the problem. There is no need to configure your environment to adopt your algorithm, just know a little bit about the syntax. It is an interactive matrix calculator, with a full-fledged programming environment. *Matlab* allows to handle with general tasks and very delicate purpose. The main focus with is what you need to program not how to program it.

Matlab is an easy to use environment, it is a fourth-generation programming language¹ (4GL). It is a very high level language.

¹http://en.wikipedia.org/wiki/Fourth-generation_programming_language



Desktop layout

Like any other environment, *Matlab* has a menubar and a toolbar where user can find the major commands needed to configure his/her preferences, this means how to let *Matlab* appear or behave. Then, the main interface is split into four majors areas as shown in FIG. ??.

Area 1

it is the Command Window, where user can strike the Matlab commands and see results displayed on the same screen.

Area 2

It is the Workspace. All variables are saved in this area. If you need to know more about saved variables, just try this command on area 1:

```
1 >> whos
```

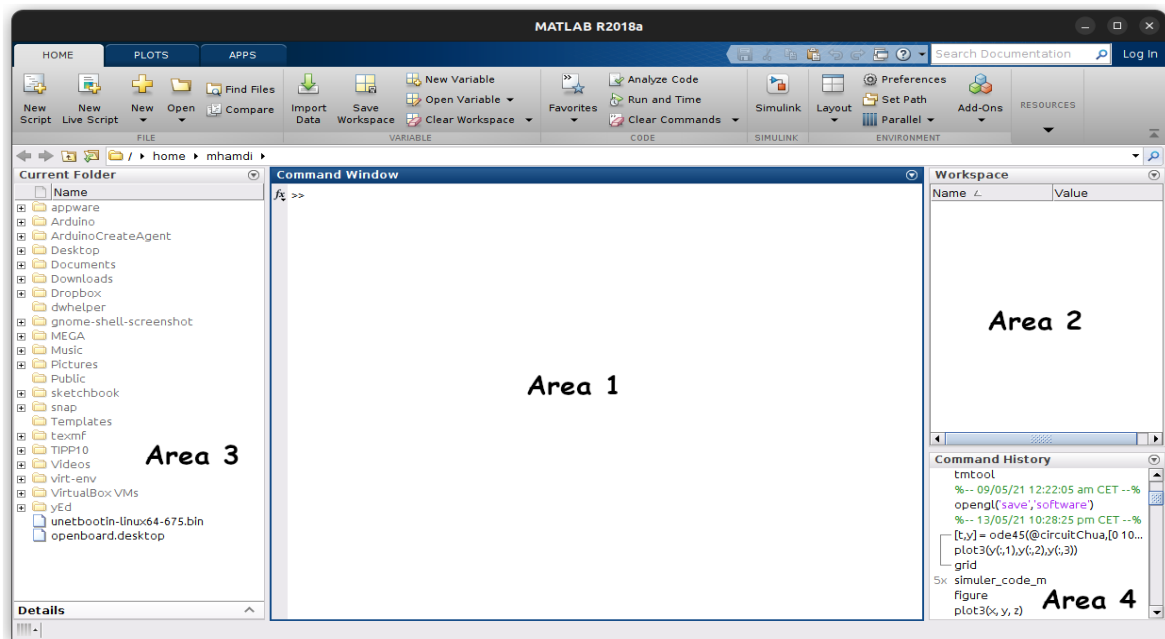
The results are the names of variables, the min and max values for every variables, class and the number of bytes needed to save it.

Area 3

It is the Current Folder, the path indicated by Matlab to execute a particular code. if you need to execute an algorithm which is not on the path shown by are 4, you have then to change the folder. Otherwise, an error message will pop-up on command window if it is not the absolute path.

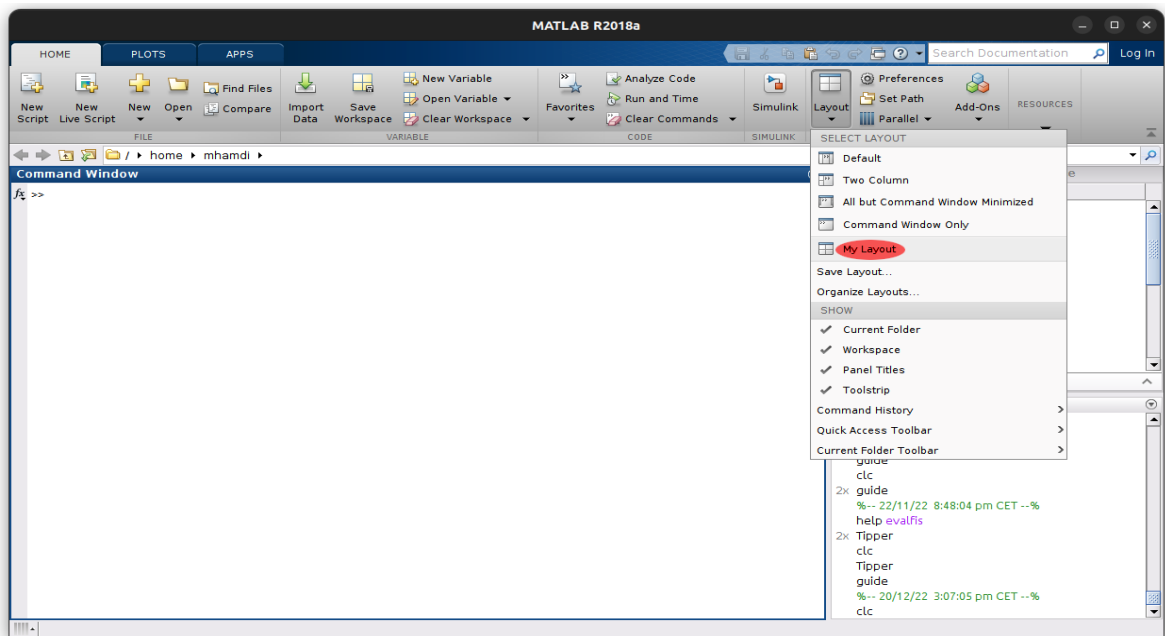
Area 4

It is the Command History, where all instructions are saved in a panel from the date of installing Matlab on your machine until current time.



How to personalize the desktop layout

In the menu bar, click on **Layout** then load your preferences. For example, here the customized layout was saved under the name My Layout. You just click on it, and your layout appears like the display shown on the following image.




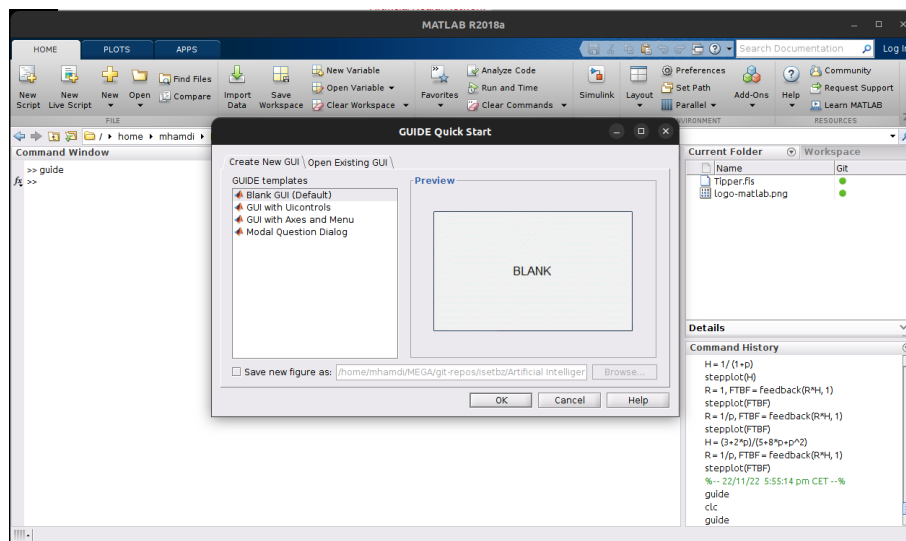
Matlab is not just an environment for technical computing, where we can solve equations. It is also an environment of graphical interfaces development. This integrated software is called "**GUIDE**", Graphical User Interface Development Environment.

Try, at the command window, the instruction:

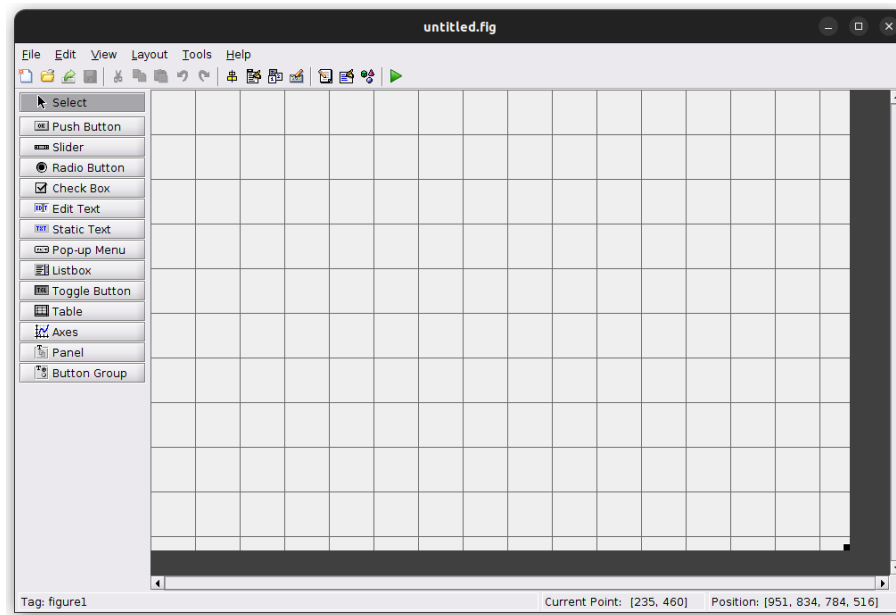
```
1 >> guide
```



or simply move your mouse on the toolbar, you will have a shortcut like the icon . Just click on it, you will then launch the environment of graphics. You will get on your screen, the dialog shown here. You can access the templates or simply click next.



Now, the environment is ready to be used.

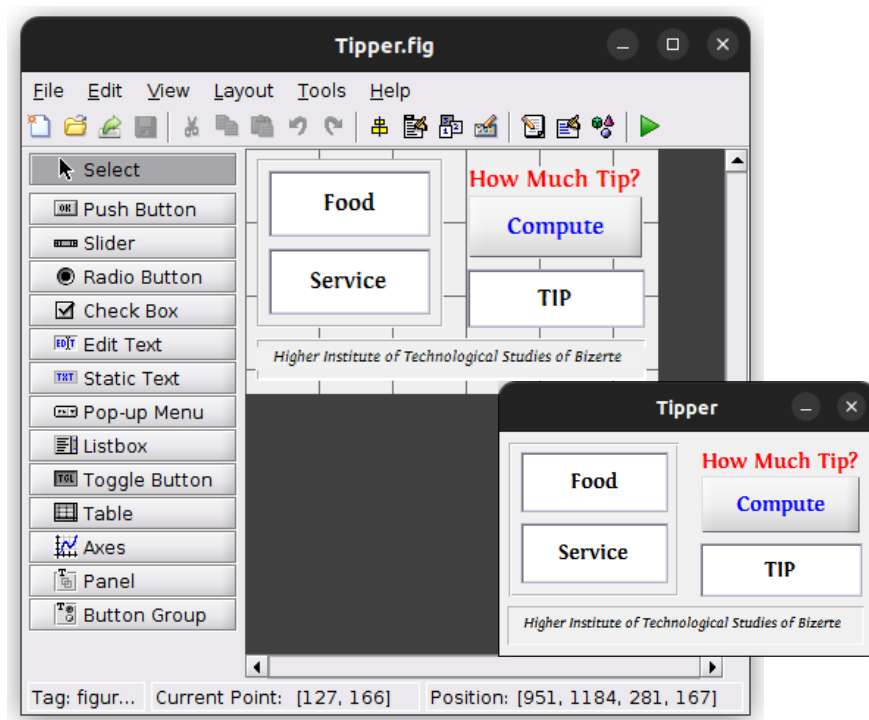


Save your work, you will notice the creation of two files on your directory with the same name, with two different extensions, one has this termination *.fig* and the second one is an *m-file*.

The Fig-file contains the graphical declaration or the instantiation of all the used graphical objects in your interface. The main GUI that appears to the user in order to customize his interface contains a palette where you can easily access the preconfigured objects like “pushbutton”, “toggle button” and “slider”. You can even add some external objects via the activeX control which is not in our current scope. Try to place and organize the required objects.

Notice that you have for any object an Object Browser which, when pointed on an object, lists all properties relative to this one and even the set of methods called when evoking it. You may also need to know that there exist a *uipanel*, use it in case you need to regroup elements by similarities.

Now, you can launch your application. Your conceived interface may look like this one indicated below.



It is preferable to load the fuzzy inference system, denoted hereafter by *fis*, in the opening function of the gui.

```

1 % --- Executes just before Tipper is made visible.
2 function Tipper_OpeningFcn(hObject, eventdata, handles, varargin)
3 % This function has no output args, see OutputFcn.
4 % hObject    handle to figure
5 % eventdata  reserved - to be defined in a future version of MATLAB
6 % handles    structure with handles and user data (see GUIDATA)
7 % varargin   command line arguments to Tipper (see VARARGIN)
8 fis =readfis('Tipper.fis');
9 handles.fis = fis;

```

Before, quitting the function, you have to update the GUI and save all variables in the general structure with handles and user data².

```

1 % Choose default command line output for Tipper
2 handles.output = hObject;
3
4 % Update handles structure

```

²Visit: <http://www.mathworks.com/help/matlab/gui-building-basics.html>

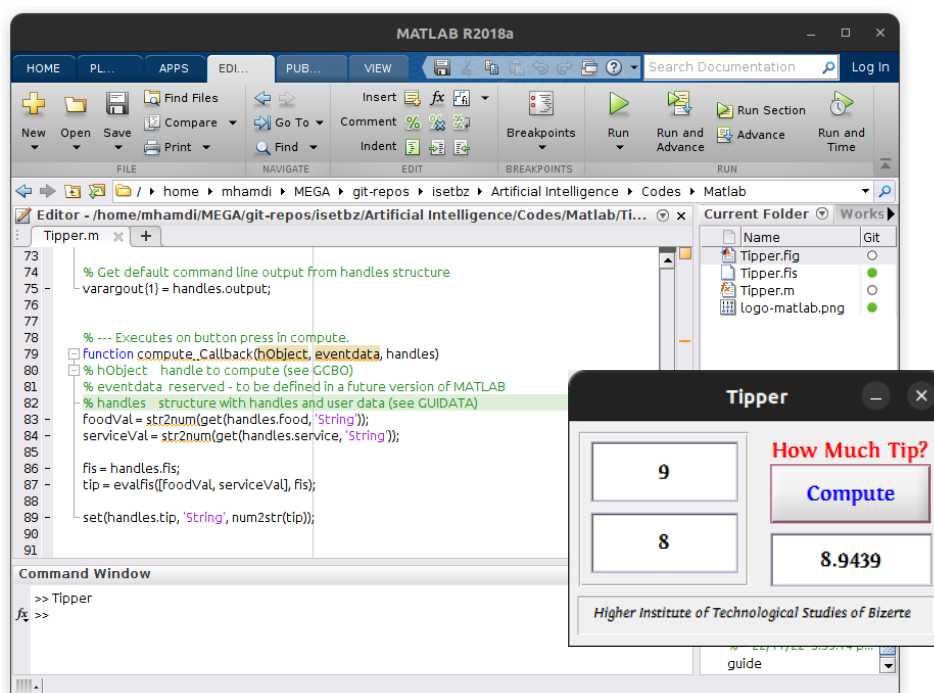
```

5 guidata(hObject, handles);
6
7 % UIWAIT makes Tipper wait for user response (see UIRESUME)
8 % uiwait(handles.figure1);

1 % --- Executes on button press in compute.
2 function compute_Callback(hObject, eventdata, handles)
3 % hObject    handle to compute (see GCBO)
4 % eventdata  reserved - to be defined in a future version of MATLAB
5 % handles    structure with handles and user data (see GUIDATA)
6 foodVal = str2num(get(handles.food, 'String'));
7 serviceVal = str2num(get(handles.service, 'String'));
8
9 fis = handles.fis;
10 tip = evalfis([foodVal, serviceVal], fis);
11
12 set(handles.tip, 'String', num2str(tip));

```

The ultimate application will behave like the example displayed in the following figures. Enter your desired values and then just click “**Compute**”. The result will be displayed on right half of the interface.





The notebook is available at <https://github.com/a-mhamdi/jlai/> → Codes → Julia → Part-2 → tipper.jl

```

1  using Fuzzy
2  using Plots
3
4  score = range(0, 10, length=100)
5
6  food = Dict(
7      "Rancid" => TrapezoidalMF(0, 0, 2, 4),
8      "Delicious" => TrapezoidalMF(6, 8, 10, 10)
9  )
10 food_chart = chart_prepare(food, score)
11
12 service = Dict(
13     "Poor" => TrapezoidalMF(0, 0, 2, 4),
14     "Good" => TrapezoidalMF(3, 4, 6, 7),
15     "Excellent" => TrapezoidalMF(6, 8, 10, 10)
16 )
17 service_chart = chart_prepare(service, score)
18
19 tip = Dict(
20     "Cheap" => TrapezoidalMF(0, 0, 1, 3),
21     "Average" => TrapezoidalMF(2, 4, 6, 8),
22     "Generous" => TrapezoidalMF(7, 9, 10, 10)
23 )
24 tip_chart = chart_prepare(tip, score)
25
26 rule_1 = Rule(["Rancid", "Poor"], "Cheap", "MAX")
27 rule_2 = Rule(["", "Good"], "Average", "MAX")
28 rule_3 = Rule(["Delicious", "Excellent"], "Generous", "MAX")
29
30 rules = [rule_1, rule_2, rule_3]
31
32 #= GRAPHS =#
33 p1 = plot(score, food_chart["values"], ylabel="Food", label=food_chart[
34     ↪ "names"], legend=:bottomright)
35
36 p2 = plot(score, service_chart["values"], ylabel="Service",
37     ↪ label=service_chart["names"], legend=:bottomright)
38
39 p3 = plot(score, tip_chart["values"], xlabel="Score", ylabel="Tip",
40     ↪ label=tip_chart["names"], legend=:bottomright)

```

```
38
39 graphs = plot(p1, p2, p3, layout=(3, 1), lw=2)
40 savefig(graphs, "mf-graphs.pdf")
41
42 # FUZZY INFERENCE SYSTEM: MAMDANI
43 fis = FISMamdani([food, service], tip, rules)
44 eval_fis(fis, [9., 8.]
```

3 | Selection Process - Case of RAIA

Student's name

Score /20

Detailed Credits

Anticipation (4 points)
Management (2 points)
Testing (7 points)
Data Logging (3 points)
Interpretation (4 points)

Goals

- ★ Design a fuzzy system to pick the most adequate candidates out of the applicants to the master program **RAIA**.



The notebook is available at <https://github.com/a-mhamdi/jlai/> → Codes → Julia → Part-1 → selection-process.jl

Fuzzy logic is a type of logical system that allows for the representation and manipulation of uncertainty in a formalized way. It is often used in the design of control systems because it is able to handle the imprecision and uncertainty that is often present in real-world systems.

By using fuzzy logic to represent and manipulate uncertainty, you can design control systems that are able to handle imprecision and adapt to changing conditions.

You are asked to design an algorithm that allows picking the best candidates out of the applicants applying for the master program **RAIA**.

In order to keep things simple, we were given only two criteria to judge:

1. their score when submitting their application;
2. their score in the interview.

BUILTIN MEMBERSHIP FUNCTIONS

Using the '*Fuzzy.jl*' package, there are five types of fuzzifiers:

Triangular fuzzifier

```
TriangularMF(l_vertex, center, r_vertex)
```

- '`l_vertex`', '`center`', and '`r_vertex`' are the vertices of the triangle, in order.

Gaussian fuzzifier

```
GaussianMF(center, sigma)
```

- '`center`' is the center of the distribution;
- '`sigma`' determines the width of the distribution.

Generalised Bell fuzzifier

```
BellMF(a, b, c)
```

- '`a`', '`b`', and '`c`' are the usual bell parameters with '`c`' being the center

Trapezoidal fuzzifier

```
TrapezoidalMF(l_bottom_vertex, l_top_vertex, r_top_vertex, r_bottom_vertex)
```

- '`l_bottom_vertex`', '`l_top_vertex`', '`r_top_vertex`' and '`r_bottom_vertex`' are the vertices

Sigmoid fuzzifier

```
SigmoidMF(a, c, limit)
```

- '`a`' controls the slope;
- '`c`' is the crossover point;
- '`limit`' sets the extreme limit.

4 | Fuzzy Control of an Articulated System

Student's name

Score /20

Detailed Credits

Anticipation (4 points)
Management (2 points)
Testing (7 points)
Data Logging (3 points)
Interpretation (4 points)

Goals

- ★ Design and implement a fuzzy regulator to control the position of a disk.



The repository is available at https://github.com/a-mhamdi/disk_and_beam/

To use fuzzy logic in a control system, you will need to follow these steps:

1. Define the input variables for the system, such as the current temperature or the speed of a motor.
2. Define the output variables for the system, such as the desired temperature or the power supplied to the motor.
3. Define fuzzy sets for each input and output variable. A fuzzy set is a set of values with a degree

of membership ranging from 0 (not a member) to 1 (fully a member).

4. Define the fuzzy rules that describe the relationships between the input and output variables. These rules should be written in the form of "if-then" statements, such as "if temperature is hot then power is high".
5. Use the input variables and fuzzy rules to compute the output variables using a fuzzy inference system.
6. Defuzzify the output variables to obtain crisp (numeric) values that can be used to control the system.

5 | Exploring the fundamentals of ANN

Student's name

Score /20

Detailed Credits

Anticipation (4 points)
Management (2 points)
Testing (7 points)
Data Logging (3 points)
Interpretation (4 points)

Goals

- ★ Dipping one's toes into the world of artificial neural networks (ANNs);
- ★ Demonstrate the ability of ANNs to learn and generalize from training data.



The notebook is available at <https://github.com/a-mhamdi/jlai/> → Codes → Julia → Part-1 → xor-gate.jl

The XOR (exclusive OR) gate is a useful example to study when learning about artificial neural networks (ANNs) because it is a simple yet non-linear function that cannot be represented by a single perceptron, which is a basic building block of ANNs.

By studying the XOR gate, students can learn about the limitations of single perceptron models and how they can be overcome by using multiple perceptrons or other types of ANN architectures. Additionally, the XOR gate is often used as a benchmark test for evaluating the performance of different ANN models, so it is useful to understand how it works and what challenges it presents.

6 | Binary Classifier using ANN

Student's name

Score /20

Detailed Credits

Anticipation (4 points)
Management (2 points)
Testing (7 points)
Data Logging (3 points)
Interpretation (4 points)

Goals

- ★ Train an artificial neural network to predict and classify categorical outcomes.



The notebook is available at <https://github.com/a-mhamdi/jlai/> → Codes → Julia → Part-1 → classifier-ann.jl

A classifier using artificial neural networks (ANNs) is a machine learning model that is trained to identify the class or category of an input based on certain features or characteristics. ANNs are a type of neural network that are inspired by the structure and function of the human brain, and they are commonly used for tasks such as image and speech recognition, natural language processing, and prediction.

In the case of a classifier, an ANN is trained on a labeled dataset, where each input has a corresponding class label. The classifier learns to map the input features to the correct class label by adjusting the weights and biases of the connections between the neurons in the network. Once trained, the classifier can make predictions on new, unseen data by applying the learned mapping to determine

the most likely class label for the input.

7 | Project Assessment

The final project will offer you the possibility to cover in depth a topic discussed in class which interests you, and you like to know more about it. The overall goal is to provide you with a challenging but achievable assessment that allows you to demonstrate your knowledge and skills in fuzzy logic or neural networks.

Some possible topics for fuzzy logic could include fuzzy set theory, fuzzy rule-base systems, and applications of fuzzy logic in control systems. For neural networks, possible topics could include feedforward neural networks, convolutional neural networks, and applications of neural networks in image classification and natural language processing.

You have to provide all necessary resources, such as sample code, relevant datasets, as well as creating a set of slides to present your work. You are expected to demonstrate your understanding of the material covered throughout this course by reviewing the basics of fuzzy logic or neural networks, as well as familiarizing yourselves with relevant programming languages and libraries. The final project is comprised of:

1. proposal;
2. report documenting your work, results and conclusions;
3. presentation;
4. source code (*You should share your project on **GITHUB**.*)

PROJECT PROPOSAL

It is about two pages long. It includes:

- Title
- Datasets (*If needed!*)
- Idea
- Software (*Not limited to what you have seen in class*)
- Related papers (*Include at least one relevant paper*)
- Teammate (*Teams of three to four students. You should highlight each partner's contribution*)

PROJECT REPORT

It is about ten pages long. It revolves around the following key takeaways:

- Context (*Input(s) and output(s)*)
- Motivation (*Why?*)
- Previous work (*Literature review*)
- Flowchart of code, results and analysis
- Contribution parts (*Who did what?*)

Typesetting using \LaTeX is a bonus. You can use **LyX** (<https://www.lyx.org/>) editor. A template is available at <https://github.com/a-mhamdi/jlai/tree/main/Codes/Report>. Here what your report might contain:

1. Provide a summary which gives a brief overview of the main points and conclusions of the report.
2. Use headings and subheadings to organize the main points and the relationships between the different sections.
3. Provide an outline or a list of topics that the report will cover. Including a table of contents can help to quickly and easily find specific sections of your report.
4. Use visuals: Including visual elements such as graphs, charts, and tables can help to communicate the content of a report more effectively. Visuals can help to convey complex information in a more accessible and intuitive way.



If you used Julia, you can generate the documentation using the package **Documenter.jl**. It is a great way to create professional-looking material. It allows to easily write and organize documentation using a variety of markup languages, including **Markdown** and \LaTeX , and provides a number of features to help create a polished and user-friendly documentation website.

I will assess your work based on the quality of your code and slides, as well as your ability to effectively explain and demonstrate your understanding of the topic. I will also consider the creativity and originality of your projects, and your ability to apply what you have learned to real-world situations. I also make myself available to answer any questions or provide feedback as you work on your projects.

The overall scope of this manual is to introduce **Artificial Intelligence (AI)** , through either some numerical simulations or hands-on training, to the students enrolled in the master's program **RAIA**.

The topics discussed in this manuscript are as follow:

① Getting started with *Julia*

Get familiar with *Pluto* Notebook.

② Fuzzification, inference system & defuzzification

Membership functions; COG.

③ System control using fuzzy logic

④ How to build an ANN

Julia; *REPL*; *Pluto*; *Fuzzy*; *Flux*; *CUDA*; *Matlab*; artificial intelligence; system control; fuzzy; inference; ann.