SmartLabDetection - Fuzzy Inference System

Daniel Lopes

Department of Electrical Engineering,
Computer Engineering & Informatics

July 5, 2022

1 Objective

During the worse times of the pandemic, Técnico imposed a limit of 2 persons inside a certain lab. However, the students that use the lab had frequent deadlines, and often ignored the 2-person limit. Having this lab configured with many sensors, a good amount of data was collected over a period of time. The objectives of this second part of the project is to apply the knowledge about fuzzy systems to the described problem with the objective of finding out if students are ignoring the limit. It is clear that this system will not have the same kind of performance as neural networks but it will give a good idea on how fuzzy systems behave and how to implement them.

2 Data Preparation

As with all data science techniques, data preparation is one of the most important steps, even if we're using different systems like Fuzzy Systems. In the case of Fuzzy this is even more important because it increases the need of feature selection to avoid combinatorial rule explosion and lack of interpretability.

Following the same approach of data preparation as with the first part of the project and given the nature and source of the data, a simple drop of all the rows that contained at least one missing value was performed.

Note that not removing missing values would hinder the system as there would need to be rules that contemplated these cases and even so it wouldn't make sense to test the system with incomplete examples..

Outliers however needed to be dealt with and, for that purpose, the technique used in the previous part of the project was still adequate as it did not remove too many data points.

Having that in mind, the standard deviation technique was applied to all elements that differed $k * \sigma$ from the average (with k = 6), which was able to detect and remove 4 outliers.

Finally to prepare the data set for the problem at hands, the feature **Persons** was transformed into a boolean class (True or False) on whether the number of Persons inside the lab was greater than 2.

2.1 Feature Selection & Feature Extraction

In order to create efficient and effective rules, one must look at the data and find out the most relevant features. In some cases it might even be beneficial to create new features that provide more precious information.

At first, it seemed obvious that one feature had to be included in this data set: **hours**. This feature would facilitate the creation of rules that consider the time of the day. This

is useful because if you look at the data, more people are in the lab in the morning or afternoon and less will be at night, even more so in the time between midnight and 8am, where there is a very small probability that more than 2 people will be inside.

The second step in reducing the dimensionality of the data was to combine some the features into a single one. Looking at the correlation matrix, this becomes even more obvious since there are groups of variables the appear to have similar correlation to the **Persons** feature.

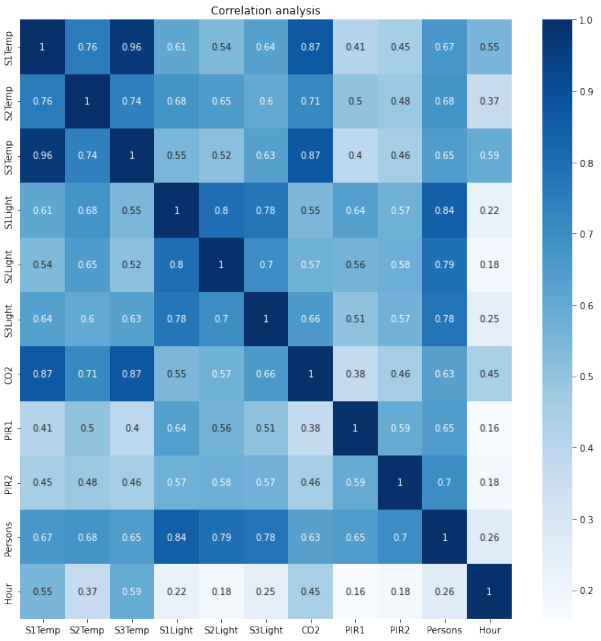

Figure 1: Correlation Matrix

With this said, the new set of features of the dataset were:

- **CO2**
- **Hour**
- **Temp** (replaces S1Temp, S2Temp, S3Temp by their average)
- **Light** (replaces S1Light, S2Light, S3Light by their sum, in order to still be able to assert how many lights are on)
- **Motion** (replaces PIR1, PIR2 by the result of PIR1 && PIR2 - logical operator)
- **CO2_Diff** (the difference between the current CO2 value and the one found 10 data points prior)

Also note that after some tuning, the **Temp** and **Motion** features were removed as they didn't add any value to our predictions and would render useless combinations of rules. The final structure of the set was actually:

- **CO2**
- **Hour**
- **Light** (replaces S1Light, S2Light, S3Light by their sum, in order to still be able to assert how many lights are on)
- **CO2_Diff** (the difference between the current CO2 value and the one found 10 data points prior)

3 Experimental Setup

With the purpose of avoiding overfitting of any kind, the data set was split into two different sets, one for validation and another for the final test which would indicate real world performance.

The validation set contained 80% of the full training data, but given the fact that this is not neural networks a split of 50/50 would've been just fine. Nonetheless, this was the chosen approach as it gave more confidence when tuning the rules and features.

Following the split of the data, the objective of the following sections was to create a fuzzy system and tune it as best as possible by removing any unnecessary feature, or by tuning the created rules (which can be found in the file *best_rules.csv*.

4 Fuzzy System

This next section describes all of the steps performed with the purpose of creating a fuzzy system and also the tuning steps that went into this process.

4.1 Fuzzy Variables

The first step in creating a fuzzy system is defining the Fuzzy Variables, i.e., creating the **antecedent** and **consequent** objects for the *ControlSystem* (class of Skfuzzy).

With a thorough analysis of the extremes of all of the features, one was able to generate the antecedents and consequent variables. After that and with some tuning behind it, the membership functions of each of the antecedents and consequent variables were created. Most of these functions were manually tuned to provide the best value. The tuning process will be described in the following sections.

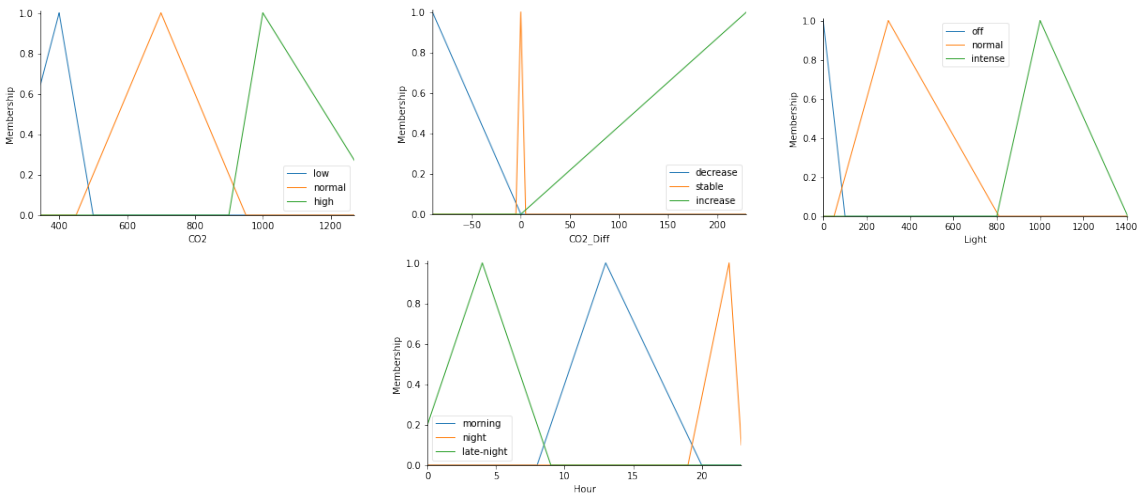

Figure 2: Membership functions

4.2 Fuzzy Rules

At the start, creating this rules manually in code was a nightmare. It proved to be unreadable, hard to maintain and hard to write. For this reason, a more thoughtful approach was taken.

Having in mind that it's always useful to have a layer of abstraction between the person who writes the rules and the person who implements the system, by writing the rules on an excel sheet, or in this case a csv file, the programmer would be able to abstract from the rules and simply read them from the file. That's what was done, a simple function that reads a csv file containing rules and instantiates a **Rule** object from the skfuzzy package.

The **csv** file contained the following format:

Hour	CO2	Light	CO2_Diff	Persons
0-23	Low, Normal, High	Normal, Intense, Off	Decrease, Stable, Increase	True, False

Table 1: Rules CSV Format

Later on, this proved to be the best helper in tuning the system as it facilitated the modification of rules.

4.3 Prediction System

Finally after reading and parsing the rules, the control system was instantiated with the Skfuzzy ControlSystem object.

With this system it was then possible to create a Control System Simulation that receives inputs and is able to compute a **CRISP** value based on fuzzy logic, for the given inputs. To run and validate this system on any data set, a helper function was created which also evaluates and returns the prediction scores of that data set.

4.4 Tuning

This step is where the magic happened. With a lot of trial and error, using the validation set, the first conclusion that was taken, which was already mentioned to give context, was that the features of temperature and motion could be removed and would not hinder the results. On the contrary, this actually benefited the system.

The last step was to improve the rules and find the most generic ones that could describe the problem. At first it seemed reasonable to look at all of the combinations of inputs but this quickly turned out to be impossible to manage. More generic rules would be absolutely necessary and that's where the focus was for the entire development of the rules. These rules can be found in the file: *best_rules.csv* on the data directory of the project.

4.5 Evaluation

After all of the tuning evaluated with the validation set, which resulted in a more optimized set of features and rules, it was time to run the system on the holdout test set which contained 20% of the original data set.

Bearing in mind that the maximum F1-Score obtained with the validation set was 0.7254, the test set did not disappoint and the results were very similar which tells us that the system is generalizing the data that we have, reasonably well.

The full results of the test set, including the confusion matrix, were:

Metric	Value
Accuracy	0.9531
Recall	0.7688
Precision	0.6796
F1-Score	0.7214

Table 2: Fuzzy System Metrics for Test Set

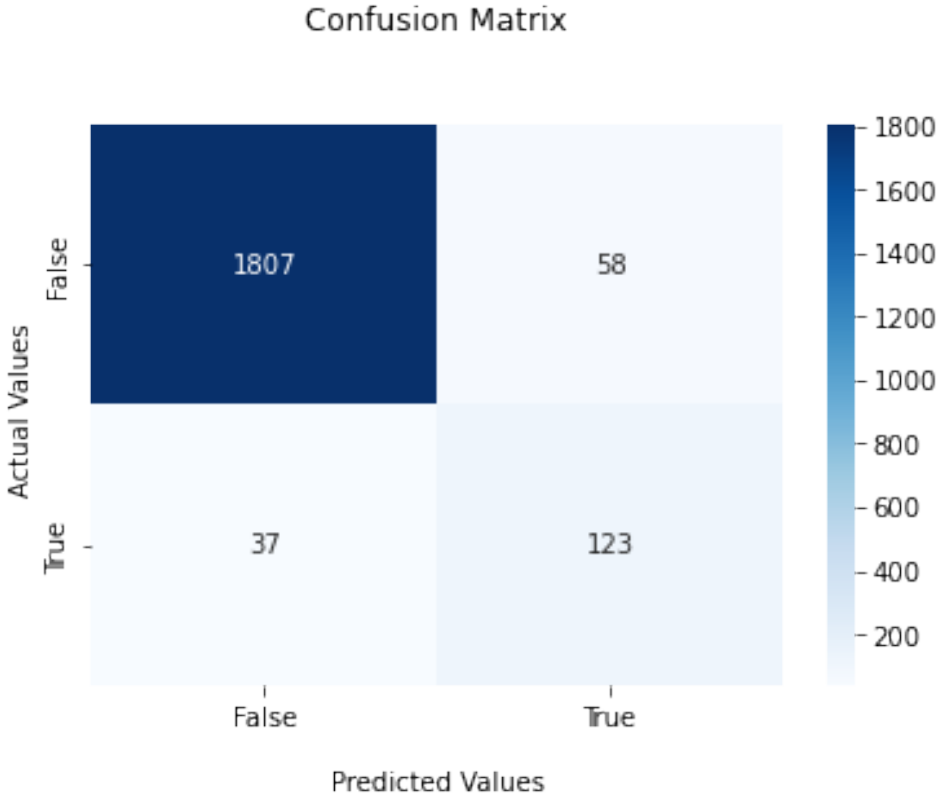

Figure 3: Results on Test Set with Fuzzy System