Project 2 – Fuzzy Systems and Evolutionary Optimization

Each record corresponds to a 20-minute period of operation of the device that contains two inputs and one output:

* Input 1: The normalized number of requests received by the device during the 20 min period.
* Input 2: The processor average load during that period.
* Output: Normal Operation (0) or Crash (1).

The first task was to develop a fuzzy system with rules based on the expert’s suggestions

[Fuzzy systems]

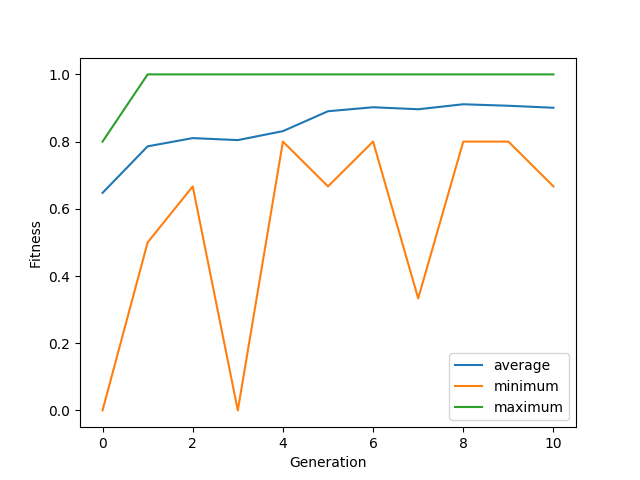
The second task was to optimize the hyperparameters from the neural network developed on the first project using an evolutionary algorithm. For this the deap library was used.

This time, to improve the results from the last project, new features were added:

* ‘High\_Load’ – binary value, 1 if the load value exceeds 0.53
* ‘High\_features” – binary value, 1 if the load value exceeds 0.53 and the requests value exceeds 0.2
* ‘Requests\_mm’ – Moving average with size 3 for the request’s values
* ‘Load\_mm’ – Moving average with size 3 for the load values

Cenas

After running the algorithm, the results were extracted.



From the graph it is possible to conclude that the evolutionary algorithm is working. The maximum fitness remains at the maximum value for almost all the run, indicating that the best individuals are affecting the next generations and being used for mating. The average trend has an upward trend for most of the run and the minimum trend is quite variable, mostly caused by poor performing individuals caused by mutations.

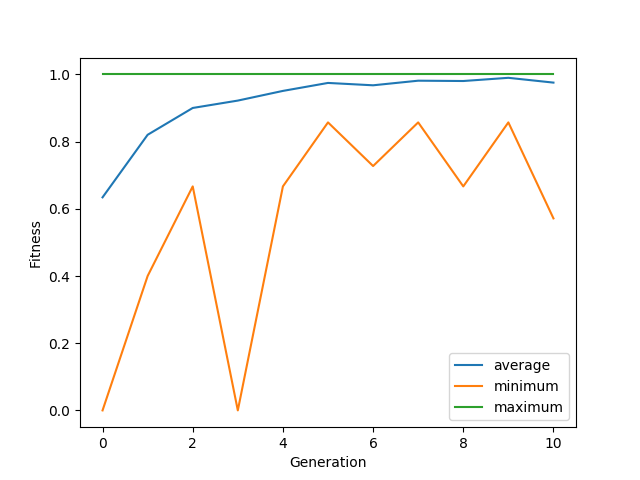
The best configuration (best fitness) was as follows:

* Layer sizes:
  + (100, 23, 153)
* Input features
  + ['Requests1', 'Load', 'High\_requests', 'Requests\_mm']

This configuration yields the following results:

|  |  |
| --- | --- |
| * Training set:   + F-measure: 1.0   + Confusion Matrix:   [490 , 0]  [0 , 8] | * Testing set:   + F-measure: 1.0   + Confusion Matrix:   [495 , 0]  [0 , 4] |

A new run was made with oversampling enabled to see if it would give the same results:



The best configuration (best fitness) was as follows:

* Layer sizes:
  + (198, 233, 58)
* Input features
  + ['Requests', 'Requests2', 'Load', 'High\_load', 'Requests\_mm', 'Load\_mm']

This configuration yields the following results:

|  |  |
| --- | --- |
| * Training set:   + F-measure: 1.0   + Confusion Matrix:   [490 , 0]  [0 , 8] | * Testing set:   + F-measure: 1.0   + Confusion Matrix:   [495 , 0]  [0 , 4] |

Generalizations

Generalization LTFS

This configuration yields the following results:

|  |  |
| --- | --- |
| * Training set (old dataset):   + F-measure: 0.57   + Confusion Matrix:   [965 , 2]  [593 , 394] | * Testing set (new dataset):   + F-measure: 0   + Confusion Matrix:   [173 , 0]  [6 , 0] |

Generalization MLP

This configuration yields the following results:

|  |  |
| --- | --- |
| * Training set (old dataset):   + F-measure: 0.99   + Confusion Matrix:   [982 , 3]  [0 , 985] | * Testing set (new dataset):   + F-measure: 0.95   + Confusion Matrix:   [189 , 1]  [0 , 9] |

Generalization ea

This configuration yields the following results: (no oversampling)

|  |  |
| --- | --- |
| * Training set (old dataset):   + F-measure: 0.92   + Confusion Matrix:   [983 , 2]  [0 , 12] | * Testing set (new dataset):   + F-measure: 0.9   + Confusion Matrix:   [188 , 2]  [0 , 9] |

This configuration yields the following results: (oversampling)

|  |  |
| --- | --- |
| * Training set (old dataset):   + F-measure: 0.99   + Confusion Matrix:   [983 , 1]  [0 , 985] | * Testing set (new dataset):   + F-measure: 0.82   + Confusion Matrix:   [186 , 4]  [0 , 9] |

*Conclusions*

On this project, the analysis of the problem had the help from some experts that gave some suggestion on how to approach the problem and had inside knowledge on the behavior of the gateways. With this the approach was more methodical and structured and the results were getting better as more advices were considered.

Although the approach that seemed more applicable for this problem, the LSTM, was used it did not present very significant results and using an MLP with features that were constructed from previous values of the request values was proven produce better results.

Adding a feature that indicated the instances where the requests were too high, improved the model. The idea of adding a fuzzy classifier to choose whether the requests were too high was not implemented due to time restraints but would make the model more robust and would work better on other datasets.