Master in Informatics and Computing Engineering
3rd year - Artificial Intelligence

Neural networks for pulsars identification

Authors:
Daniel Filipe Santos Marques - up201503822
Pedro Miguel Ferraz Nogueira da Silva - up201505460
Renato Alexandre Sousa Campos - up201504942

April 12, 2018

Abstract

This project is being developed as proposed by the professors in the Artificial Intelligence class of the Master's degree in Informatics and Computing Engineering at the Faculty of Engineering of the University of Porto.

The progress in developing better telescopes has led to them being capable of gathering enormous amounts of data. The objective of this project is to process this data, automating the identification of Pulsars using Artificial Neural Networks. These algorithms are inspired by real neurons and, like them, they learn with specific situations, improving its accuracy over time.

1 Description

1.1 Specification

The program must train a Neural Network capable of classifying stars as Pulsars or Non-Pulsars, making use of the "Back-Propagation" algorithm. The data set is available courtesy of UCI's machine learning repository. The data set is composed of 8 statistics variables and the resulting classification. The first 4 variables are obtained from the integrated pulse profile and the remaining 4 variables from DM-SNR curve. Luckily, this data has been preprocessed for machine learning.

A Neural Network is multi-layered: the input layer is made up of 8 neurons that correspond to the 8 variables from the data set and the output layer has only 1 neuron which corresponds to the classification of the star (0 for negative, 1 for positive). Additionally, there should be at least a hidden layer for the NN to learn properly. As of this writing, we've found the balance between execution time and accuracy to be a single hidden layer with 100 neurons, though this can be submitted to change as more testing and research is done.

There is a lot to tweak when it comes to NN's hyper-parameters and the right tunning can give an enormous boost in both accuracy and efficiency. Such hyper-parameters include: the number of hidden layers and their number of hidden neurons and implementation dependent parameters.

For development the data set will serve 2 use cases: training (80%) and testing (20%). The first is needed for the network to learn (tweak its weights

and biases) while the second will provide valuable information regarding the accuracy and evolution of the training.

1.2 Implementation progress

Neural Networks have increased in popularity the last few years and therefore it's not difficult to find information and implementations to get started on this topic. However, we've chose that instead of using a high-level artificial intelligence framework we'd develop from a native language with the objective of learning more by delving into the code ourselves, which is easier with a smaller codebase. With this in mind we've decided to read the online book "Neural Networks and Deep Learning" by Michael Nielsen, understand the provided code and modify it to our needs.

The code example is almost what we needed aside of some differences. The example used a Neural Network to identify hand-written digits. Because of this, there were multiple (10) output neurons (one for each digit) with the result being the index of the neuron with the highest activation. Our case is quite different since the expected result is binary (either a 0 or a 1). With this is mind we've decided to consider an activation that's less than 0.5 to be a 0 and 1 otherwise. In addition, the format of the file itself required a different loader and data structure conversion. Having done this our neural network became fully functional and, at that time, running with an average of 98.00% accuracy.

Finally, two problems can demonstrate very different accuracy values under the same hyper-parameters and architecture. Therefore, there must be extensive testing on this matter to achieve the best accuracy and performance. We've done some research on how to choose and tune these values but there's still a lot to test.

1.3 Expected results and evaluation

Once properly trained, a Neural Network is capable of evaluating new cases and learn even more. However our data is limited, therefore we chose to use 80% of it to train the network and 20% to test it, demonstrating that it indeed does work as expected. Our program will show statistics regarding the training phase such as accuracy and cost during epochs, if such option is selected, or will run silently until the testing phase where it'll show those statistics only regarding the test data. Due to the uneven

percentage of positive pulsar cases in the data set, these statistics should also be divided into one per class (pulsar or non-pulsar), demonstrating that the neural network isn't biased towards the majority (non-pulsar). Testing a Neural Network like ours means to run the feed-forward function which will not affect the network's weights or biases but will return the neuron's activations (for each test case).

The expected result is to have the highest accuracy possible in the test phase (which tries to simulate new cases) with the lowest cost possible. It's also very important to monitor the cost during training to ensure it decreases steadily which is a good sign the neural network is learning correctly as opposed to it being stagnant or over-fitted.

2 Conclusion

So far, we've understood how a Neural Network works and adapted an existent implementation to our specific problem, meeting our objective of efficiently automating the process of identifying pulsars. We believe projects like these with even more advanced techniques will help our civilization evolve by either accelerating our discoveries or solving complex social problems. Either way, there's no doubt there's a place for Artificial Intelligence to be used in every scientific field if people are aware of its limitations.

3 Resources

Nielsen, Michael. 2017. *Neural Networks and Deep Learning.* neuralnetworksanddeeplearning.com.

Data set: https://archive.ics.uci.edu/ml/datasets/HTRU2.