Artificial Neural Networks: Perceptron

First Project Report

Carl Habib – Elie Azar

Abstract—This report serves to document the creation of an intelligent software agent: the perceptron. Designed based on user characteristics, the perceptron then goes through a training phase, in order for its outputs to be validated according to user requirements. This perceptron serves as the main building block in hand-written character recognition software by having the intelligent agent to extract features that help in identifying properties that are used for character recognition.

– – – – – – – – – ◆ – – – – – – – – –

Contents

1 Introduction

THIS paper will be discussing the first out of a two-part project in the Intelligent Engineering Algorithms course. Throughout this paper, we will be explaining how a perceptron works and how it could be implemented in real life applications. It details the development, training, and testing of both a single and a multi-layered perceptron.

2 Background

2.1 Context

Machine Learning is a method of data analysis that gives computers the ability to learn without being explicitly programmed. This field has been exploited for more than half a century, and scientists and data analysts' work has come into fruition recently, especially with the rise of Artificial Intelligence and Machine Learning in our daily lives: These methods manifest in automatic data classification, accurate web search, self-driving cars, practical speech, handwriting recognition, checkers and backgammon games…etc. This field has proved its usefulness in human life in general. Machine learning is generally classified into three categories: Supervised learning, Semi-Supervised learning, and Unsupervised learning. Though they all achieve the same purpose, each of these categories does so differently: Supervised learning uses labeled training inputs in order to determine a certain output, while Unsupervised learning aims to reach the output while devising structures using unlabeled data sets. Finally, Semi-Supervised learning falls in between the aforementioned categories: it determines an output based on partial training with limited data sets.

Neural networks are the backbone frameworks for machine learning algorithms, and they form the core of this project. Artificial Neural Networks (ANN - which are interconnected multi-layered perceptrons) have been designed since the latter half of the 20th Century. The development of such structure was inspired by the work of scientists Warren McCulloch and Walter Pitts. The importance of such structure is due to the fact that ANNs are used for building mathematical brain models that simulate the activity of biological brains, a scientific breakthrough that helps understanding the inner workings of the human brain. The perceptron is a learning intelligent agent, which means that it adapts to change by retaining previous data, all while being noise resistant. For these reasons, machine learning is one of the most important computer schemes that help achieve human-like level of Artificial Intelligence.

3 PROPOSAL

3.1 CONCEPTS AND BUILDING BLOCKS

The intelligent agent in this project was designed to correspond to the PEAS conceptual model that describe: i) the environment, 22) its sensors, iii) its actuators, and iv) its corresponding performance function.

I. Design & PEAS Model:

In order to build the perceptron, we start by defining the environment in question. Hence, in this project, following the corresponding PEAS conceptual model of the perceptron, we start by defining each aspect:

- Performance: the accuracy criterion which evaluates the performance of the perceptron at hand.
- Environment: scenarios based on which the data sets will be defined
- Actuators: the outputted trained sets that will be used to validate the user's input
- Sensors: classifiers that are used to classify the data based on trained data

II. Functions:

The customized perceptron was made to be as generic as possible. It can support both modes: single layered as well as multi-layered. As previously learnt in the IEA course and following the algorithm's steps we implemented this function to account for the interlayer output.

$$\mathbf{y_j} = \boldsymbol{f}_{\mathbf{w},\boldsymbol{\theta}}(\mathbf{X}) = \boldsymbol{f}_{\mathbf{j}}(\mathbf{W}^{\mathbf{T}}\mathbf{X} - \boldsymbol{\theta}_{\mathbf{j}}) = f_{\mathbf{j}}\left(\left(\sum_{i=1\ldots I} \mathbf{w}_{\mathbf{ij}} \times \mathbf{x}_{\mathbf{i}}\right) - \boldsymbol{\theta}_{\mathbf{j}}\right)$$

The following functions were made available to define the function per cell:

- Gaussian
- Sigmoid
- Unity
- Bounded Linear

The following functions were made available to define the error function per cell:

- MAE
- MSE
- MDE

III. Configuration:

Within the first panel of the interface, the user is presented with a multitude of functional options: He could load previously configured perceptron models, in order to test or train them.
When choosing to create a perceptron, the user chooses the number of input and output cells, while the hidden layers and their neurons have to be created manually. On top of that, he should set the error threshold based on one of three available error functions. The weights could be either automatically generated, or manually set. The perceptron can be saved with the click of a button, with all of its data being saved in an XML file. Multiple perceptrons (with their respective layers, cells, error and activation functions) are saved in the same file.

IV. Training and validation:

After the perceptron 's creation, the training phase debuts with the input data and desired output provided by the user. The whole concept revolves around updating the weights as much as needed so that the actual output fits the desired output

V. Scenarios:

Scenario 1: Random Quadrant Classifier

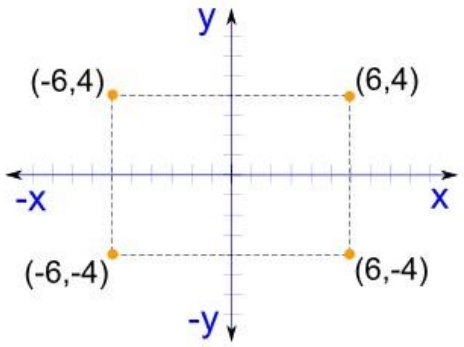

Figure 1: Quadrant Classifier

This first scenario consists of training the perceptron to classify points with (x, y) coordinates into their corresponding quadrant.

Scenario 2: Even and Odd Classifier

The second scenario trains the perceptron to take two number inputs, and determine whether their addition is even or odd.

3.2 DESIGN

The project was designed in order to appeal to as many scenarios as possible, which gives the user freedom to fit any custom build. The very basic component of this project is the cell, the neuron. Without it, making connections that give us an intelligent agent would not be possible. Cells can be arranged in layers, depending on whether we have a multiple input or outputs. The two classes above make use of many different

mathematical classes we have created in order to compute outputs: the matrix class allows matrix calculations, while the functions and error machine classes allow us to set activation and error functions in every cell in the perceptron.

The aforementioned entities form the perceptron, which is modeled in its own class, and offers training options, with randomly generated data. The generated perceptrons may be saved in a single XML file in order to be used again at a later time at the user's convenience.

The whole is modeled as follows:

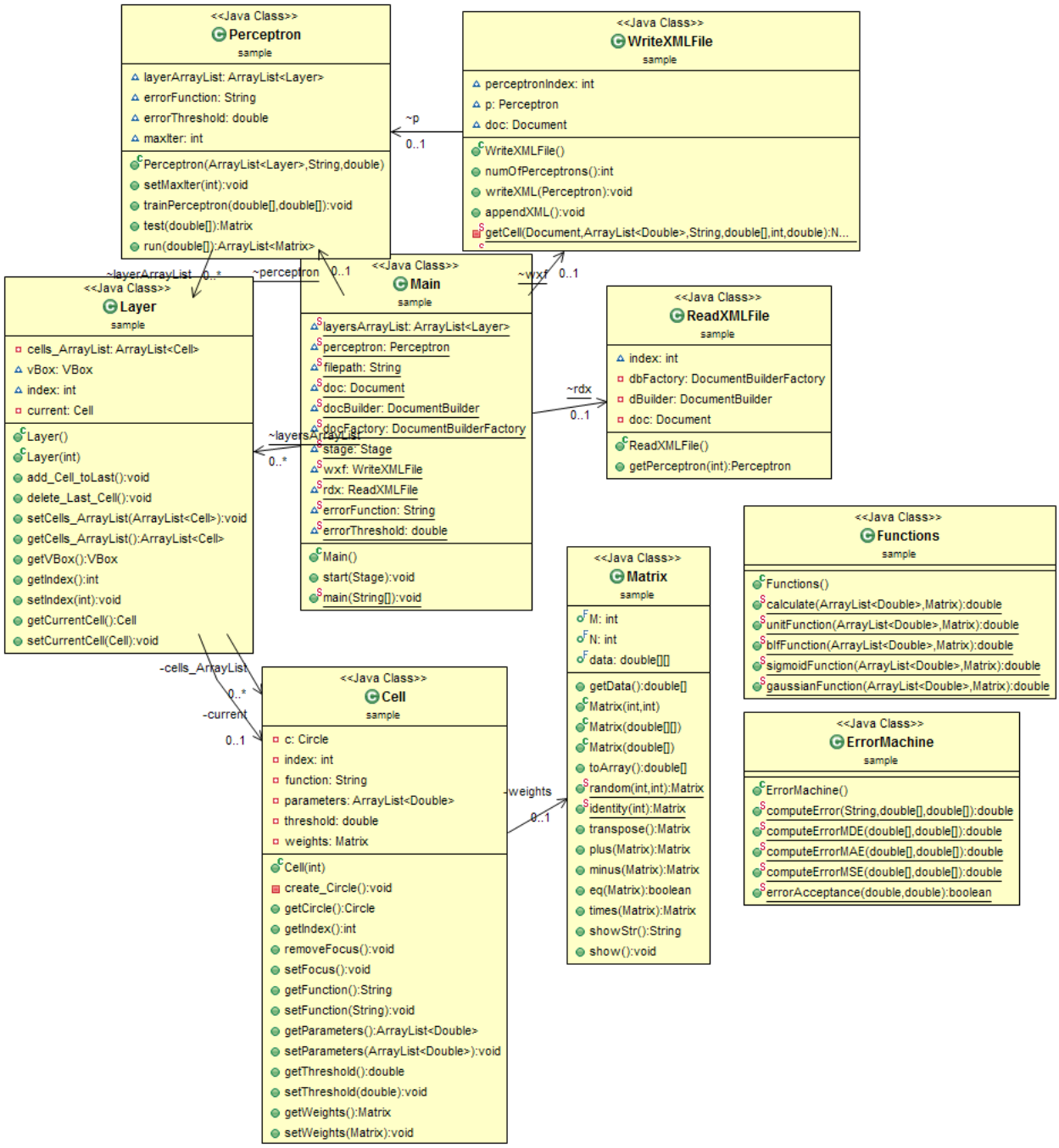

Figure 2: UML Class Diagram

3.3 TESTING AND FINDINGS

This final phase concerns evaluating the proposed design. The user can use the either the quadrant classifier or the Even-Odd classifier (with test data points) in order to see how the perceptron works without being trained at first, and how training affects the overall outputs.

We found that the perceptron was only about 50% accurate according to heuristics, and this is due to different factors. Most times, the errors can be attributed to the underfitting element: updating weights in the perceptron happened in a diffident manner; it thus took many trials and training attempts for the perceptron to get simple outputs right.

Another great difficulty we faced was getting the right initial perceptron weights. And this was a substantial factor in deciding how the perceptron would behave, and the training required for it to present accurate answers later on.

4 CONCLUSION

This project was an insightful take regarding the creation, training, and manipulation of an artificial neural network. The testing phase showed us where the perceptron works accurately at times, and where its shortcomings are visible.

To conclude, we learned that the more complex a perceptron is built, the higher amount of connections, the more accurate the results will be. But in all cases, we are dealing with a computerized agent, and faulty outputs and errors are always prevalent, especially since this Artificial Neural Network is very small in scale compared to the monstrous supercomputer technology used in large corporations.

5 REFERENCES

[1] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5(4):115–133, 1943.

[2] Dawson, M. R., & Gupta, M. (n.d.). Probability matching in perceptrons: Effects of conditional dependence and linear non separability.
Retrieved from: http://journals.plos.org/plosone/article?id=10.1371%2Fjournal.pone.0172431

[3] Nielsen, M. A. (1970, January 01). Neural Networks and Deep Learning.
Retrieved from : http://neuralnetworksanddeeplearning.com/chap1.html