A Deep-learning approach to Thyroid disease classification

full name:	Endri Kastrati
student id:	24064580
coordinator:	A.B.M Russel
date:	31/08/2017
word count:	1665

Abstract

This report begins with an introduction to the proposed project. A proper description of the functional and non-functional requirements is initially outlined. It continues on by providing a detailed project plan and then elaborate on the external and internal designs as well as the software architecture of the proposed project. Finally, we layout a testing plan for the proposed project and provide a conclusion to this report.

Table of contents

1.0 Introduction

The deployment of machine learning algorithms to classify medical conditions is a relatively new field of computer science that has begun very recently to gain widespread support. Medical institutions as well as governments have accumulated fast quantities of medical data that are rich both in features and content. Because of the relevance and intellectually challenging aspects of such problems, I have taken the initiative to develop an artificial neuron network that learns from a large set of thyroid disease data and can predict future unseen datasets with a high degree of accuracy.

2.0 Project Requirements

The project aims to solve the degree of uncertainty in regards to thyroid gland disease classification by providing the medical expert or practitioner with a reinforced confidence about the final diagnosis stage. Specifically, its purpose is not to replace the medical experts but rather enhance their diagnostic skills.

2.1 Functional Requirements

There are five main functionalities that the project must abide by. These core functional requirements are mandatory for the correct implementation of the project.

- The computer program must be able to accept a large dataset as input and fetch it into the neural network. The neural network must be able to run the back-propagation algorithm to adjust its synaptic weights in order to generalize the classification/fitting patterns.

- The computer program must be able to dump the adjusted synaptic weights of the neural network into a directory in binary format.

- The computer program must be able to load the adjusted synaptic weights into a neural network and use them to forward propagate given signal inputs for classification/fitting.

- The medical expert must be able to use an application interface to provide a collection of input data (features) which will be fetched into the neural network as input signals.

- The medical expert must be able to retrieve the output of the classification query in a user-friendly format.

In terms of the application interface, that could potentially take the form of a web application or a desktop graphical interface depending on the needs and requirements of the medical expert/practinioner.

2.2 Non-Functional Requirements

Performance: The training process of the neural network has to be as fast as possible and consume a minimum amount of memory such that it can be deployed into a microcontroller or an embedded device with limited memory capabilities.

Portability: The project should be able to compile and run on different platforms without any bottlenecks or incompatibility issues. This property of course depends heavily on the choice of programming language and numerical analysis package.

Security: The core program, namely the artificial neuron network processes, should contain no warnings,error messages or memory leakages when run.

Reusability: Certain parts of the project, such as the ANN library should be flexible and portable as a standalone package for other learning purposes such as curve fitting or time variant systems.

Extensibility: The core parts of the project should be designed in such a way that enables the implementation of extensions or different training algorithms for the neural network.

3.0 Project Plan

Due to the complexity of the project a thorough and detailed plan has to be outlined in order to identify the main objectives, requirements, constraints, risks and risk mitigation techniques.

3.1 Overview

The first phase of the project life-cycle development includes the investigation and research on the topic of connectionist systems, more specifically artificial neural networks and their algorithmic and mathematical analysis. A proper type of neural network has to be selected as a prime candidate for the physical implementation process. The first milestone is the implementation of an abstract library that provides an API for interacting with the neural network data structure. Second milestone is the development of a working prototype that abides by the function requirements of the project. The final milestones include testing, a final version of the project and a thorough evaluation of the project life-cycle development process.

3.2 Risk Analysis

Below is risk register table that elaborates on the different types of risk manifestations, the corresponding probability occurrence and impact for each risk and the risk mitigation techniques

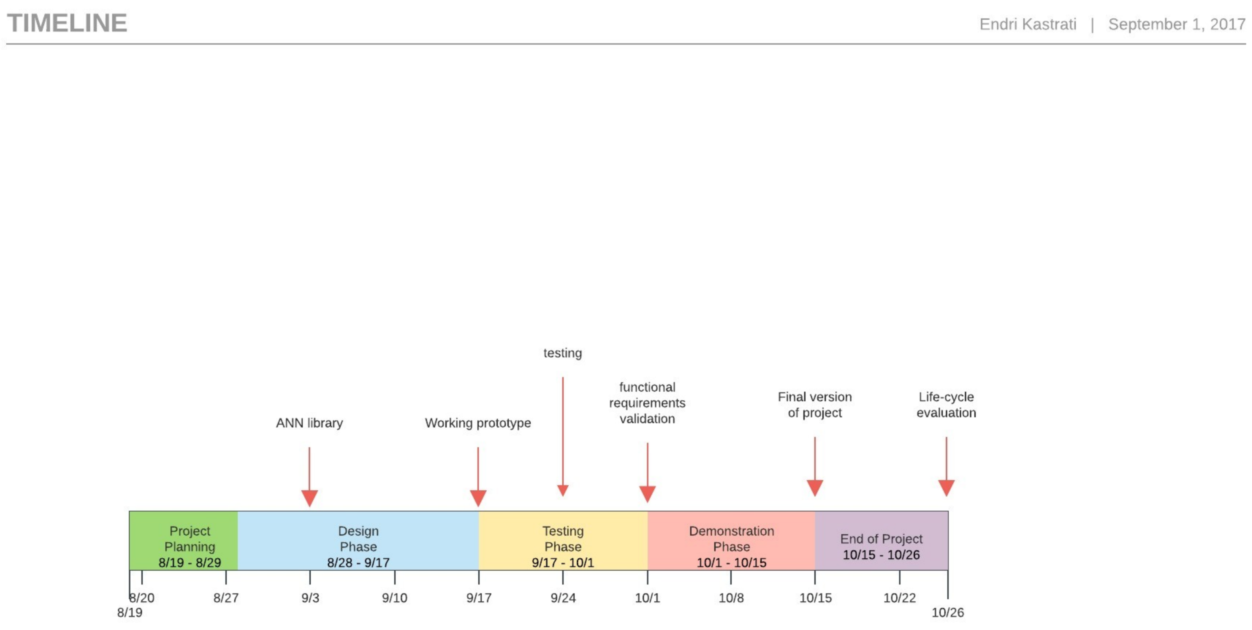
Risk	Description	Category	Triggers	Root cause	Potential responses	Risk Owner	Probability (high 5, low 1)	Impact (high 5, low 1)	Score
ANN does not work	the back propagation algorithm Is not working as expected.	Software	improper behaviour of the program output	Inability to understand and properly implement the back Propagation algorithm.	Re-design or fix the back-propagation algorithm	Developer	4	5	20
API does not work	The Application interface does Not working as expected	Software	improper behaviour of The IO interface	bugs in the API code and Horribly designed interface	Tweak the interface Module to fix bugs	Developer	3	3	9
ANN takes too long To adjust weights	The neural network training Process is taking too long to adjust the synaptic weights	Software/ Hardware	Penalises productivity And efficiency for the Medical expert	low computational power, No optimization whatsoever	Re-design the bottlenecks And optimize the process Of the back-prop algorithm	Developer/ User	1	2	2
Non-existent GSL lib	The host computer does not Have the GNU scientific library Installed locally.	Software/ Hardware/ User	Inability to compile and Execute the aplication	User did not install project Dependencies	Install the GNU scientific Library and re-compile	User	4	3	12
Horrible user-interface	The user interface is so complicated Or confusing that the user is unable To query for a classification result	Design/ Software	Inability to run the Application program	Not properly taking into Consideration usability Engineering principles	Re-design the user-interface And accept user feed-back	Developer	4	4	16

3.3 Resource Requirements

A 64-bit computer, the GNU C compiler and the GNU scientific library are required for the core parts of this project. In regards to the application interface, possible candidates includes GTK3 for a desktop GUI or Nodejs/HTML/CSS for a web interface. Because of the minimalistic nature of the project these libraries suffice for the entire duration of the project development life-cycle. The Cmocka testing framework will be used for the testing phase of the project and GNU profilling and the valgrind memory checker will de employed for performance analysis and memory management.

3.4 Schedule

The following timeline diagram demonstrates the expected schedule and core milestones of the project development life-cycle.



4.0 External Design and Internal Design

There are three main interfaces for the application program. The first interface is the command line program that receives a number of parameters and trains the neural network data structure using the backpropagation algorithm. The second interface involves the command line program again but this time the program should be able to load the adjusted weights into a neural network data structure and output the predicted classification values based on the given input data. The third interface involves the interaction of the medical expert/practitioner with a user-friendly program in the form of a GUI or web app. The high level user-interface should be able to receive input data from the medical expert and fetch it as arguments into the command line program. The command line program receives the given input, loads the adjusted synaptic weights and prints the classification value. The high level-interface receives the output from the command line program and prints it to the medical expert in an understandable and user-friendly format.

4.1 User Interface

The following pseudo-code provides an example of how the command line program interface should look like:

For the training process of the neural network:

```
>$ ./neuralnet --train  (  --curve-fitting | --pattern-classification ) --in-file=<filepath>
        --dump-dir=<filepath> --features=<number> --layers=<number>
        --neurons-per-layer=[ number, .. ] [--epsilon=<number>]
        [--eta=<number>]  [--epoch=<number>]
```

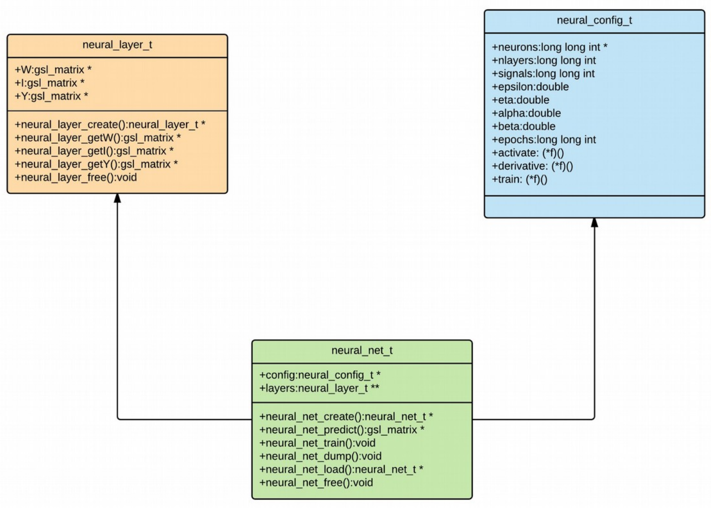
For the prediction process of the neural network:

```
>$ ./neuralnet --predict --load-dir=<filepath> --in-file=<filepath> --out-file=<filepath>
```

4.2 Performance and Memory consumption

The training process has to be extremely fast and efficient in terms of memory consumption. More specifically, the total number of memory blocks allocated should always be equal to the number of memory blocks initially required during the instantiation process of the neural network. This means that all future operations should mutate the state of the already allocated matrices and should not request additional memory from the heap. In regards, to time complexity because of the non-linear nature of the back-propagation algorithm, there is no formal guarantee on the number of iterations required per training session.

5.0 Software Architecture



6.0 Test Plan

Unit, integration and functional testing: In order to test the validity of the ANN library and it’s components the Cmocka testing framework will be used. Initially, the validity and correction of the individual components of the library will be tested and then the entire library as a whole.

Blackbox testing: For the correctness of the back-propagation algorithm, a black box testing technique will be employed where the neural network is fetched different types of datasets and the generated outputs are compared against the desired outputs. Also a cross-validation technique will be used to obtain the accuracy of the trained models.

Performance testing: The GNU profiling tool will be used to test the time performance of the ANN library and the valgrind tool for debugging memory consumption and leakages.

7.0 Conclusion

This report introduced the proposed project and established the functional and non-functional requirements. A detailed project plan was initially outlined, focusing on the main objectives, requirements, constraints and risks as well as the project timeline. It continued with a detailed description of the external and internal design as well as the software architecture of the project. Finally testing methodologies and software tools were proposed to assist in the validation process of the project’s functional requirements. The objectives and the main goal are clear and the analytical and mathematical aspects of the porposed application constitute an excellent computer science project.

8.0 Bibliography

E. A. Bender, *Mathematical methods in artificial intelligence*. Los Alamitos, CA: IEEE, Computer Society, 1998.
C. C. Aggarwal, *Data mining: the textbook*. Cham: Springer, 2016.
A. C. C. Coolen, Kühn Reimer, and P. Sollich, *Theory of neural information processing systems*. Oxford: Oxford Univ. Press, 2005.
G. C. C. Dreyfus, *Neural networks methodology and applications*. Berlin: Springer, 2005.
K.-L. Du and M. N. S. Swamy, *Neural networks and statistical learning*. London: Springer, 2016.
O. Maimon and L. Rokach, *Data mining and knowledge discovery handbook*. Berlin: Springer, 2010.
S. J. Russell, P. Norvig, and E. Davis, *Artificial intelligence: a modern approach*. Harlow: Prentice Hall, 2016.
W. Schiffmann, M. Joost, and R. Werner, *Synthesis and performance analysis of multilayer neural network architectures*. Koblenz: Univ., FB 3: Naurwiss., Inst. für Physik, 1992.
I. N. D. Silva, *Artificial neural networks: a practical course*. Switzerland: Springer, 2017.
X.-H. Yu and L.-Q. Xu, “Automatic learning rate optimization by higher-order derivatives,” *Proceedings of International Conference on Neural Networks (ICNN97)*.