Predicting ICU Admission of Patients Using Simple Neural Networks

Jakob Škornik
University of Ljubljana, Faculty of Computer and Information Sciences
Email: js6675@student.uni-lj.si

***Abstract*—This paper contains a report on an attempt at creating and evaluating models on a real-world dataset during the *Sars-CoV-2* pandemic. Diagnostic data has been obtained at Hospital Sírio-Libanês, São Paulo and Brasilia and then preprocessed and anonymized. It contains a set of patient attributes and a binary target variable, indicating whether a patient in the hospital is subsequently admitted to the ICU ward. We have designed a single-layer neural network without any help from external machine learning packages (apart from visualization). Our models are then compared to a *Random Forest Classifier*. The main goal of this work is to observe the effect of applying different mechanisms on model accuracy and to obtain a deeper perspective of how neural networks work. Code and results can be viewed on GitHub [1].**

***Keywords*—covid-19, neural network, machine learning, python, Random Forest classifier**

I. Introduction

THIS paper contains a report on the implementation and evaluation of a simple neural network. The intended purpose of this network is to train on a real-world dataset, which contains anonymized data of patients admitted to the Hospital Sírio-Libanês, São Paulo and Brasilia. The dataset contains a binary target variable that indicates that a patient has been admitted to the ICU ward.

The report is structured in the form of seven chapters. After the introduction, a brief overview of related work is presented. It includes a short description of where many of our designs were adapted from. The third chapter describes the goals of this paper in more detail. The fourth chapter further describes our input data and the process of preparing the dataset before feeding it to the algorithm. The next chapter dives into the design and implementation of our neural network and presents challenges and design choices. Finally we present the results and conclusion of the work. The seminar project is done in the context of a machine learning course at the Faculty of Computer and Information Sciences in Ljubljana.

II. Related Work

The problem of predicting covid-19 caused ICU admission has been published on the web portal *kaggle.com* [2]. Many similar submissions have been uploaded to the before-mentioned repository. The neural network has been designed after an online project called *Neural Network From Scratch* [3] which consists of a book, sample repository and even an educational video series teaching one how to implement a neural network by oneself. Unfortunately, the project is still ongoing and isn't finished yet. It provided a baseline structure that has been adapted and repackaged in a more object-oriented design.

III. Goals

A functional and competitive neural network usually contains a large set of mechanisms and techniques for it to be competitive against other machine learning algorithms. Neural networks boast high combinatorial capabilities or in other words, are capable of approximating very complex functions. However, this same capability leaves such models prone to overfitting and certain other vulnerabilities. Thus, many mechanisms are added to cover such cases.

When deciding whether to use a neural network or some other simpler algorithm, it is often hard to decide on a neural network. Implementation takes a long time, especially as each component of the network offers several different implementations and the choices you make, often have direct impact on the quality of the model. This means, that a large chunk of the development process is taken up by comparing results between different networks with different mechanisms. For example, different weight initialization techniques, different weight and bias optimizers, dropout, momentum or cross-validation...

The goal of this work is to design and implement a neural network from scratch. This however means, that some mechanisms probably won't be included due to time limitations. This should provide us with a deep understanding of how neural networks work and how different techniques impact models performance.

IV. Data

The whole process of preparing data for modeling is described in great detail here [1].

Our data has been obtained and anonymized at Hospital Sírio-Libanês, São Paulo and Brasilia, Brazil. There are around 1400 entries with 54 features. These attributes have been scaled to fit between -1 and 1 according to the *Min-Max Scaler* [4].

Group	Amount of features
Demographics	3
Grouped diseases	9
Blood results	36
Vital signs	6
Total	**54**

TABLE I: Types of features.

Entries are also divided into time windows, which enables us to try to predict whether a patient is admitted to ICU based on how long he is in the hospital at that point. Each patient can have up to five entries, depending on when he's admitted to the ICU ward.

Window	Description
0-2	From 0 to 2 hours of the admission
2-4	From 2 to 4 hours of the admission
4-6	From 4 to 6 hours of the admission
6-12	From 6 to 12 hours of the admission
Above 12	Above 12 hours from admission

TABLE II: Time windows table.

There were also several other problems. For example, there were several null values. Due to the time windows structure, it is advised in the problem repository to use forward or backward data fill. There were also several non-numeric data types. We transformed them into numeric values and one-hot encoded multi-column features.

We had to leverage the time windows structure and add a target variable that indicates whether a patient is admitted to the ICU in any further entries, as in the original dataset we can actually see in which time window admission happens.

There were also some smaller modifications to the dataset, for example: removing whitespace characters from column names, making copies of datasets for each time window, removing unnecessary columns etc.

V. Implementation

To fully leverage the *numpy* package for numerical operations in python, we decided to implement a batch neural network. This means that the whole dataset is fed to the algorithm a set number of times, instead of each individual entry by itself.

We tried to leverage object-oriented principles so that each component can be easily swapped in the initialization of the network.

Code snippet 1: Example of neural network initialization.

```
# Create a neural network
adam128_covid = BasicNeuralNetwork(
        input_size=X.shape[1],
        output_size=2,
        iterations=2500,
        logs=True,
        log_frequency=250,
        alpha=0.01,
        alpha_decay=0.0001,
        layer_size=128,
        optimizer="adam"
)
```

In the snippet 1, we can see some of the mechanisms that we implemented. From the snippet, we can see there is learning rate decay, **Adam** optimizer and some debugging mechanisms. There are however plenty of other mechanisms that one can select. For example, there are three different optimizers included: *Stochastic Gradient Descent*, *AdaGrad* and *Adam*. If we use *Stochastic Gradient Descent* we can also select to use momentum. We can tweak hidden layer size by passing an integer and every hidden layer will have the same number of neurons, however, we can also pass a list of individual sizes and thus define an uneven distribution of neurons in the network. The network has first been tested by approximating simple boolean functions, such as AND and XOR and it proved to consistently find a good approximation in just a couple dozen iterations. Since we processed the dataset to only include numeric columns, the neural network accepted the dataset without any further processing. An interesting observation that we made, was that when we ran the training process on just a couple of neurons, the network reduced the problem to the majority class.

VI. Results

We also ran the same dataset on a *Random Forest Classifier* with an 85/15 split of training and testing sets. We used these results to compare and evaluate our models.

We decided to use an 85/15 split because of a rather low amount of entries. Out of 1400 entries only about 400 were admitted to the ICU. Especially for the first couple of time windows, there was a really low amount of data. Surprisingly our neural networks outperformed the baseline model on the smallest dataset. Aside from accuracy we also calculated the sensitivity and the specificity of our models, as these two metrics are most important when evaluating medical and diagnostic models.

Let's talk about sensitivity and specificity a bit more. Sensitivity is basically just the ratio between detected positives and true positives. For example if we detect every positive outcome we would have a 100% sensitivity. If we predict positively for every entry we would still have a 100% sensitivity, but our specificity would be at 0%. Specificity is the ratio between predicted negatives and true negatives.

Now, when we are talking in the context of our problem, there are two scenarios where these two metrics become more important than the other. If we have enough beds and personnel in the hospital, we might value sensitivity more, so we don't miss any potential ICU admission, but any false positives don't bother us too much. However, if we are running low on beds, we might value specificity more. We want to filter people that really need the beds. False positives are a much bigger problem in this case.

Let's take a look at the results presented in the three tables. We can compare all three optimizers to the *Random Forest Classifier* on each dataset.

Algorithm	**0-2**	**2-4**	**4-6**	**6-12**	**All**
SGD	65%	72%	71%	72%	70%
AdaGrad	67%	73%	68%	75%	71%
Adam	57%	75%	73%	70%	70%
Random Forest	68%	78%	87%	90%	89%

TABLE III: Table of accuracies of individual algorithms.

Our algorithms *plateau-ed* at about 75% accuracy, which turned out to be competitive to the RF classifier in the first two time windows. However, with more data, present our results became noticeably worse.

Algorithm	**0-2**	**2-4**	**4-6**	**6-12**	**All**
SGD	66%	51%	48%	51%	55%
AdaGrad	66%	41%	64%	39%	57%
Adam	41%	46%	44%	66%	67%
Random Forest	57%	70%	76%	80%	82%

TABLE IV: Table of sensitivities.

The higher sensitivity of our models in the first time window just might be more important than the overall accuracy when we are screening patients at the beginning of their stay in the hospital. But as more data becomes available, once again, the RF classifier becomes much more sensitive.

Algorithm	**0-2**	**2-4**	**4-6**	**6-12**	**All**
SGD	64%	80%	80%	80%	76%
AdaGrad	67%	86%	69%	89%	76%
Adam	67%	88%	84%	71%	72%
Random Forest	80%	85%	95%	95%	97%

TABLE V: Table of specificities.

Interestingly, specificity-wise our algorithms fared much better against the RF classifier. Especially the high specificity in the time window *6-12*.

These results indicate that further testing should be done to get more accurate results. It doesn't make sense that any metric becomes that much worse with more data. A large number of models should be made and their results aggregated. This way, a much more accurate evaluation of the models can be made.

VII. Conclusion

There really isn't much innovation in this paper. The whole seminar work serves the purpose of coding practice, deepening our knowledge and understanding of neural networks. We created a batch neural network in python, added several mechanisms and techniques, such as different optimizers, momentum, learning rate decay and some others. We documented the process of preparing the data using a *Jupyter* notebook and published it on our GitHub [1]. Similar, usually better performing, models can be observed on [2], where people all over the world submitted their solutions.

The original goal of gaining a deeper understanding of neural networks was however accomplished.

References

[1] Škornik, J. Predicting ICU Admission of Patients Using Simple Neural Networks. https://github.com/JakobSkornik/covid19-admission

[2] covid-19 - Clinical Data to assess diagnosis. https://www.kaggle.com/S%C3%ADrio-Libanes/covid19, Last accessed at 1.1.2022.

[3] Neural Network From Scratch in Python https://nnfs.io, Last accessed at 1.1.2022.

[4] PATRO, S GOPAL & Sahu, Dr-Kishore Kumar. (2015). Normalization: A Preprocessing Stage. IARJSET. 10.17148/IARJSET.2015.2305.