F20BC COURSEWORK 1

Report written by Capper, David and Dymarkowski, Jakub

Introduction

Artificial neural networks are a mathematical representation of attempting to imitate a biological brain and how its neurons work (Lancashire et al., 2009). There are many types of neural networks, the one focused in this coursework is the back propagating neural network which involves a gradient-based learning algorithm and a cost (heuristic) function (Mirjalili, 2019).

Aims

The aim of this coursework:

- Implement an ANN
- Train ANN with provided dataset
- Investigate changes to hyperparameters and results
- Report investigations

Program Development Rationale

The ANN was implemented using **Python**. The packages used to complete the coursework were as follows: **math**, **numpy**, **pandas**, and **pyplot**

Note: *no ANN or AI related packages were used for implementation.*

The first iterations of the implementation involved using lists and classes to define the network, however the solution that was settled on was using matrices as it was quicker and intuitive.

The reason as to why python was chosen as the programming language of choice is due to how it's a scripting language and widely used in the artificial intelligence community as the go-to language of choice.

Another design choice was that the ANN can stop back propagating once it has reached the target accuracy, this was implemented to shorten the time taken to train and run the ANN.

The way that values are initialised in the ANN are by using a uniform randomizer. This evenly distributes the random values between -1 and 1. This technique was chosen as it would prevent the ANN from being unevenly weighted initially.

The training dataset that was provided was the Wisconsin Breast Cancer dataset.

Output.txt – holds the recent accuracy and iteration data for recently run networks. Prompted output file – holds the configurations and hyperparameters as well as time taken for the network to finish training.

Methods

The investigation of the changes to hyperparameters was conducted by having the ANN retrain with the specified configurations and inspect the output. The results are graphed, and a comparison is made which discuss the impact a change can have on the ANN.

Since there are many ways one can structure an ANN, tables are provided to show what configuration was chosen and what hyperparameters have been changed.

Values being measure:

- Accuracy
- Loss
- Time

Definitions:

- Iterations = number of times it runs one cycle of training
- Epochs = number of times the dataset is shown to the network
- Min iterations = minimum number of iterations before it exits training if the target accuracy has not been reached
- Target Acc = the target accuracy to reach when training
- Batches = number of times the training data is split

Configurations

Activation Functions: 1 – Sigmoid 2 – ReLU 3 – Tangent

The configuration tables are a representation of the structure of different ANNs

Config	Learning rate	Activation function	Number of layers	Number of node / layer
C1	0.01	1, 1, 1, 1	5	30, 20, 20, 20, 2
C2	0.001	3, 2, 1, 1	5	30, 30, 20, 20, 2
C3	0.05	2, 1, 1	3	30, 50, 10, 2

Hyperparameters

Config	Iterations	Epochs	Min iterations	Target Acc	Batches
H1	100	10	100	0.98	10
H2	100	10	50	0.95	5
H3	100	10	10	0.90	1

Results

Configuration	Hyperparameters	Time (s) / run	Ref
C1	H1	199, 147, 93, 130	Figure 1
	H2	49, 49, 48, 49	Figure 2
	H3	19, 20, 19, 20	Figure 3
C2	H1	0.85 Acc, DNF	
	H2		
	H3		
C3	H1	168, 305, 123, 111	Figure 4
	H2	62, 72, 69, 84	Figure 5
	H3	26, 27, 27, 27	Figure 6

Discussions and Conclusions

After completing all the above, there are certain conclusions that can be derived form the results and findings.

However, there are key points to discuss about the problems faced during the experimental investigation. C2 had encountered, what is believed, to be the dying ReLU problem. C2 had numerous revisions done to the ANN configuration, such as: learning rate, activation orders, number of nodes, number of layers. Throughout all the revisions, the subject of the problem seemed to have always led to involving the ReLU activation function. An implementation of the leaky ReLU was then carried out, yet that also seemed to have suffered from low values and the ANN averaged at around 85% accuracy.

The investigations have shown:

- With lower batch sizes causes the ANN to train quicker, but only with small datasets. With larger dataset using batching can improved training performance – but will lead to a “noisy” result (Smith et al., 2020).
- The number of layers and nodes an ANN has, the better it becomes at identifying complex patterns. The trade-off of this is the time it takes for an ANN to back-propagate as it has more parameters to tweak (Goodfellow, 2016).
- As stated above, not only can a larger ANN identify complex patterns but also represent functions of increasing complexity (Goodfellow, 2016).
- When the revisions to the structure of C2 where being made, changes to the learning rate were played around with. Throughout the revisions, the learning rate changed the average accuracy each run. Although C2 never reached its target accuracy, modifying the learning rate would adjust how high the accuracy would be.
- The order of activation functions greatly impacted the accuracy and values nodes had. A point of interest was to highlight was how due to the ANN working with negative numbers, having the ReLU function at the first layer cause the accuracy to be lower and error higher due to the previously stated dying ReLU problem.
- During certain re-runs and implementation testing, the average accuracy would not only have minor changes but a couple of major ones due to the randomisation of values when the ANN is initialised. This carried through into the latest iteration and design of the ANN and can be noticed when re-running. This can affect the results; this is also why multiple runs where performed.

To summarise the findings, hyperparameters have a significant impact of the ANN and its performance, while the activation functions can impact the values of the ANN, its output accuracy and loss. There is also a lot of thought that goes into designing the structure of an ANN and the problems that can arise from not considering what activation functions to use for each layer.

Appendix

Figure 1

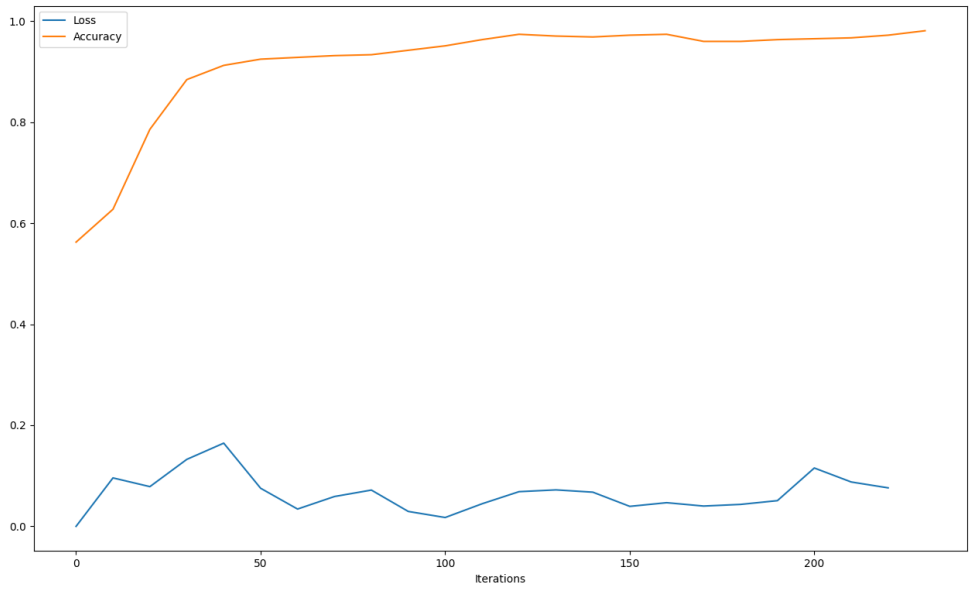


Figure 2

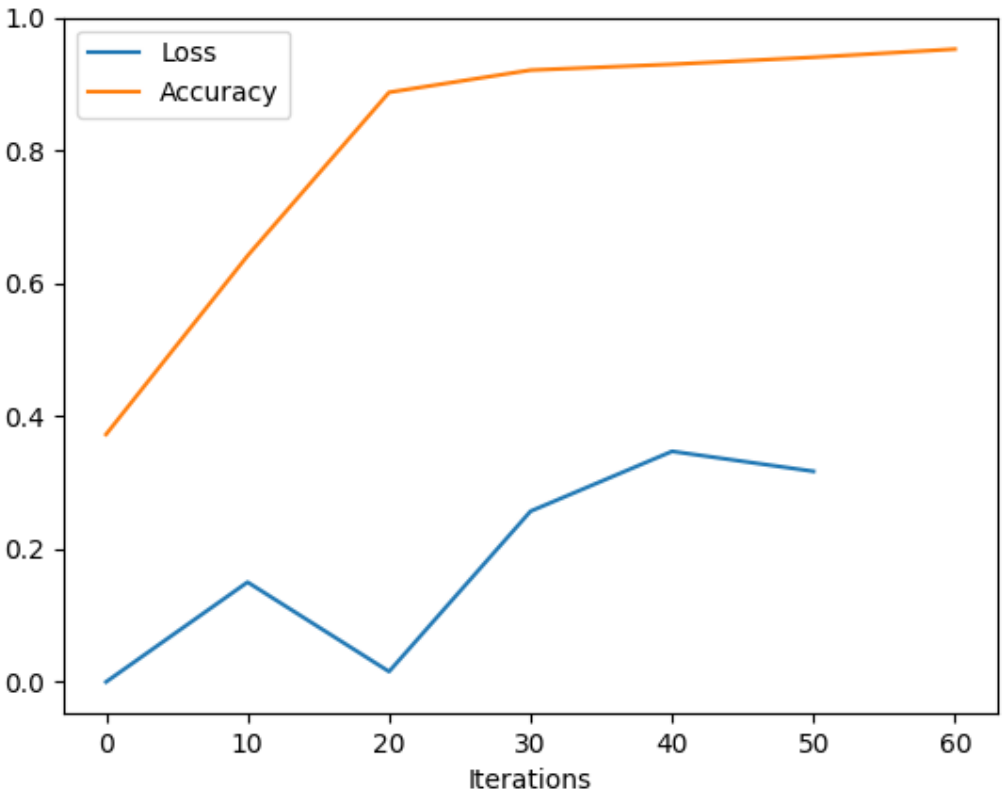


Figure 3

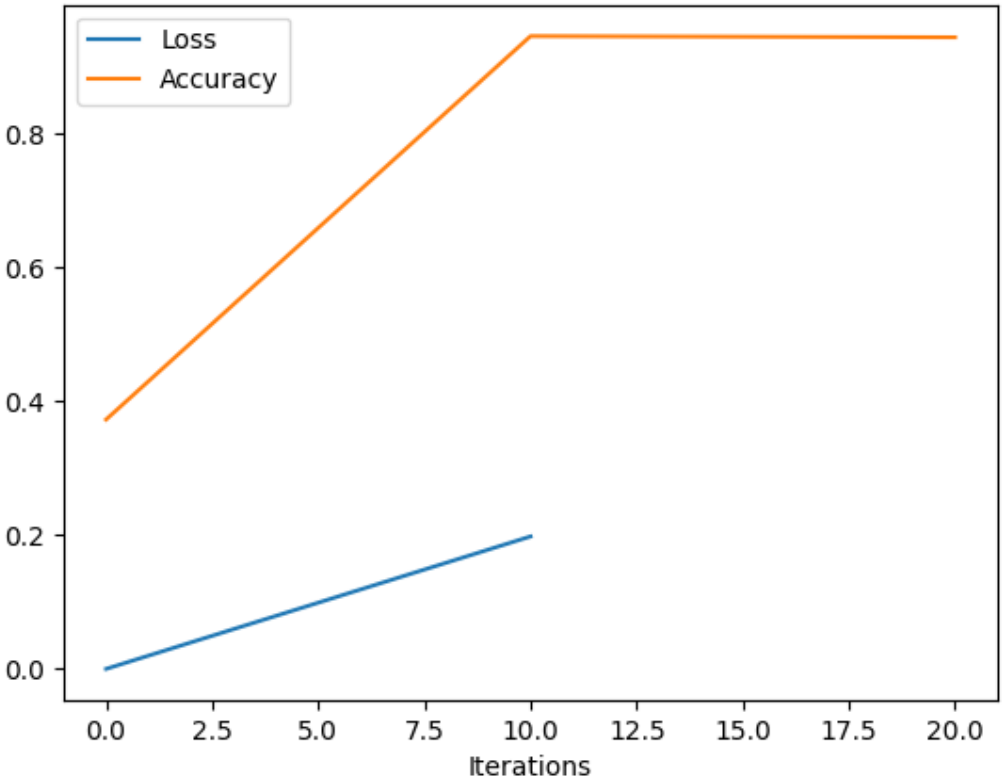

Loss
Accuracy
0.8
0.6
0.4
0.2
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Iterations

Figure 4

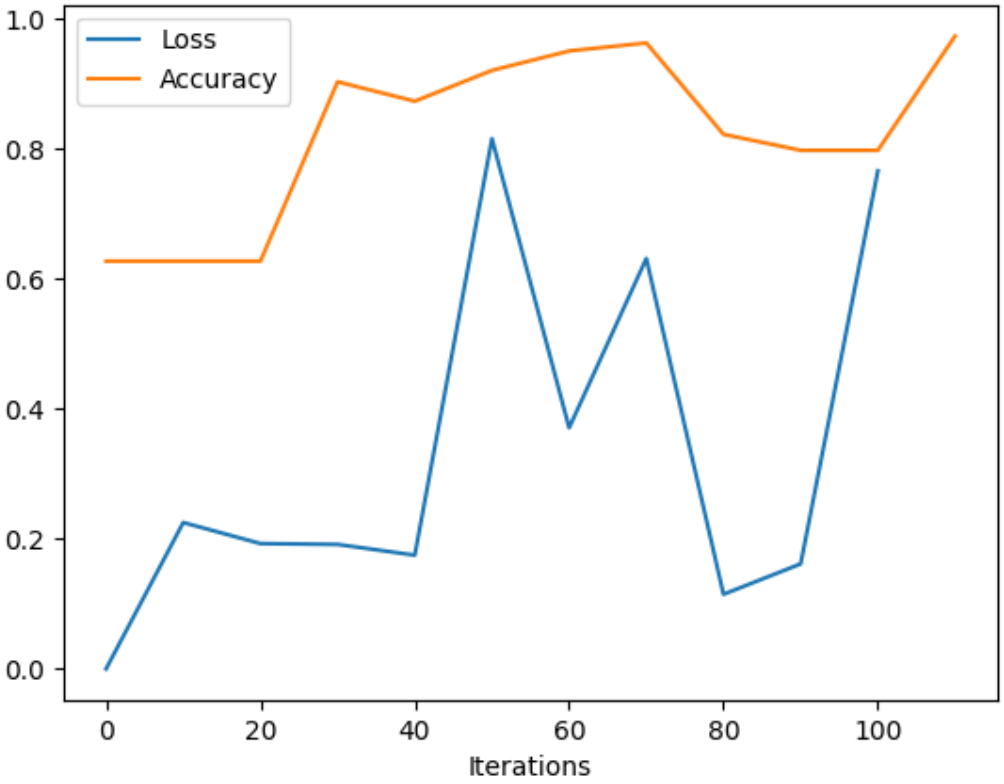


Figure 5

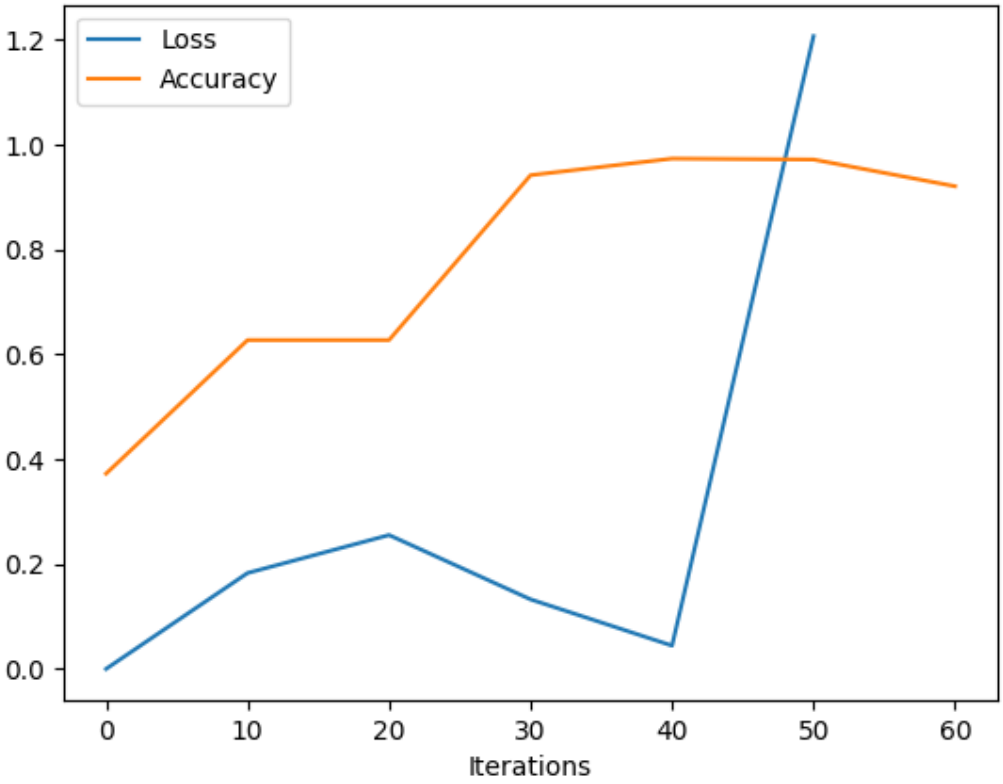
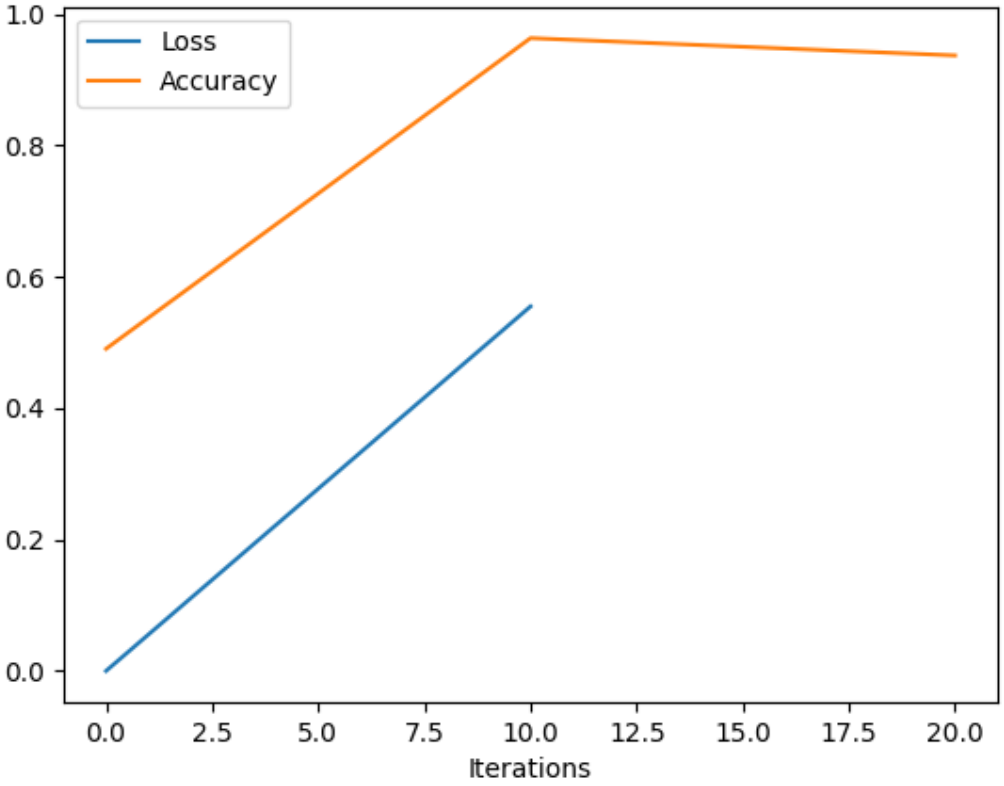


Figure 6



References

Goodfellow, I. (2016). *Deep learning / Ian Goodfellow, Yoshua Bengio and Aaron Courville.* (Y. Bengio & A. Courville, Eds.). Cambridge, Mass. : MIT P.

Lancashire, L. J., Lemetre, C., & Ball, G. R. (2009). An introduction to artificial neural networks in bioinformatics-application to complex microarray and mass spectrometry datasets in cancer studies. *Briefings in Bioinformatics*, *10*(3), 315–329. https://doi.org/10.1093/bib/bbp012

Mirjalili, S. (2019). *Evolutionary Algorithms and Neural Networks Theory and Applications / by Seyedali Mirjalili.* (1st ed. 2019.). Cham : Springer International Publishing : Imprint: Springer.

Smith, S. L., Elsen, E., & De, S. (2020). *On the Generalization Benefit of Noise in Stochastic Gradient Descent*.