UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

COS711 Assignment 2

Hybrid Learning in Neural Networks for Almond Classification

Student number: u19264047

Contents

1 Methodology

2 Data PreProcessing

2.1 Normalization

All Numbers were normalized between 0 and 1 (inclusive) to aid in the Neural Networks interpretation of the data.

2.2 Missing Values

Given the issue PyTorch has with missing values and entries containing NaN, all missing values were replaced with an indicator value, which was -1. This did not influence the range of the data normalized since it was done after normalization.

2.3 Classes

Since the Types of Almonds is nominal data binary one-hot encoding was used. The Type column was split into 3 columns. One for each potential type an almond can be, Mamra, Sanora, or Regular. Each type the almond was not was marked by a 0 and the class it was apart of was marked as 1.

2.4 Data Preparation

2.4.1 Why not remove any entries with missing values?

A reason to not remove any entries that have missing values is because none of the entries are complete, all of the entries have at least one missing value.

2.4.2 Why not remove columns with missing values?

A reason to not remove any columns with missing values is the lost of potential knowledge gain. 6 of the 12 columns have missing values. With all columns but 2 columns that have missing values have between 30% to 40%, the other 2 have 64% missing values.

2.4.3 Why not impute data?

The data can't be accurately imputed as all but the first 5 columns are values calculated by using other values in the data set. And if a value is missing all the values that are calculated with it are also missing.

2.4.4 What implications does this have for missing value imputation?

Because a missing value also means it's derived values are also missing makes it a lot less accurate to impute data.

2.4.5 Why replace with -1?

Training with missing values and an indicator for that data, helps the NN learn how to deal with incomplete data, this will help improve real world applications were not all the information is always available.

2.5 Data splits and where they are used.

The data is split into 3 sets, namely Train, Test and Evaluation. Training set consists of 50% of the data, Test set consists of 20% of the data, Evaluation set consists of 30% of the data. All entries are exclusive to one of the 3 sets.

The Training set is used in epochs for training the nn. The Evaluation Set is also used in epochs but never for training. The model is tested against the Eval set for the propose of creating graphs and measure if the model is over fitting. The Test set is used to determine how well a set of parameters performed on the model. This is explained further in the section about Results.

3 Set Up for Neural Network

3.1 Layers

3.1.1 Input Layer

This layer is always consistent, it has 12 nodes, one for each data point.

3.1.2 Output Layers

This layer is always consistent, it has 3 nodes, one for each Type. SoftMax is used as the activation function for this layers. This scales the outputs of the output layer such that they all add up to 1. Giving a result where the output is a probability of the entries being one of any of the 3 classes.

3.1.3 Hidden Layers

The number of hidden layers is one of the Hyperparameters optimized. There is a set pattern for the number of nodes in each layer. The last hidden layer always has 4 nodes, then each layer has double the number of nodes as the next. For example with 3 players the number of nodes would be 16, 8 and 4.

The activation function is the other Hyperparameter. The 5 potential activation functions and their formulas are:

1. ReLU: $f(x) = max(0, x)$
2. Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$
3. TanH: $f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
4. SoftPlus where beta was 1: $f(x) = \frac{1}{\beta} * \log(1 + e^{\beta x}) = \log(1 + e^{x})$
5. SiLU: $f(x) = x * \frac{1}{1+e^{-x}}$

4 Hyperparameter Optimisation

4.1 Parameters

As discussed in the previous section the 2 hyperparamters are number of layers and activation function.

The 5 option for Hidden layers are as follows:

1. 1 layer: 4 nodes. 4 total
2. 2 layer: 8 nodes, 4 nodes. 12 total
3. 3 layer: 16 nodes, 8 nodes, 4 nodes. 28 total
4. 4 layer: 32 nodes, 16 nodes, 8 nodes, 4 nodes. 60 total
5. 5 layer: 64 nodes, 32 nodes, 16 nodes, 8 nodes, 4 nodes. 124 total

and 5 options for activation function for the hidden layers:

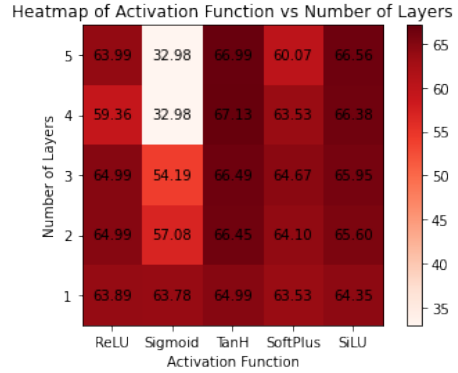
1. ReLU: $f(x) = max(0, x)$
2. Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$
3. TanH: $f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
4. SoftPlus where beta was 1: $f(x) = \frac{1}{\beta} * \log(1 + e^{\beta x}) = \log(1 + e^{x})$
5. SiLU: $f(x) = x * \frac{1}{1+e^{-x}}$

4.2 Experimental Setup

- epochs: 1000
- learning rate: 0.1%
- optimizer: ADAM optimizer
- Each hyper parameter is run 5 times and the average and standard deviation is taken.

These hyper parameters have been kept constant for the purposes of this experiment to keep everything consistent, except for the 2 hyper parameters.

4.3 Results

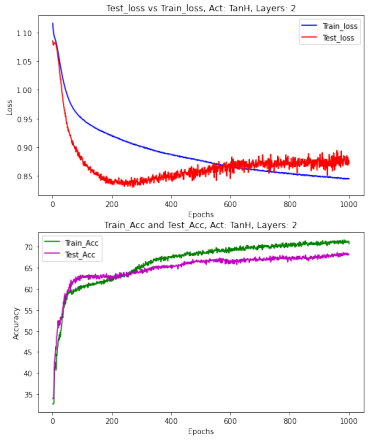


	ReLU	Sigmoid	TanH	SoftPlus	SiLU	Average	STD Dev
1	63.89	63.78	64.99	63.53	64.35	64.11	0.58
2	64.99	57.08	66.45	64.10	65.60	63.64	3.77
3	64.99	54.19	66.49	64.67	65.95	63.26	5.12
4	59.36	32.98	67.13	63.53	66.38	57.88	14.25
5	63.99	32.98	66.99	60.07	66.56	58.12	14.32
Average	63.44	48.20	66.41	63.18	65.77		
STD Dev	2.34	14.33	0.85	1.80	0.88		

4.4 Number of Layers

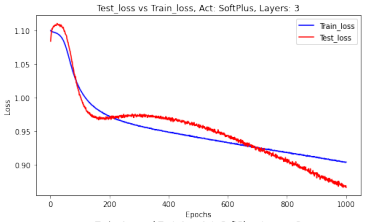
4.4.1 1 layer

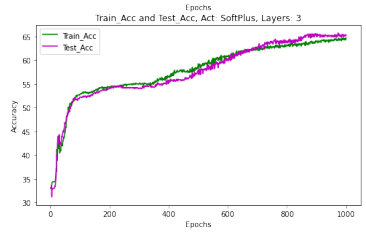
Models with 1 layer have the best average and a very low standard deviation. Models with 1 layers being the ”best” is due to Sigmoid being an outlier on larger layer models and performing very badly. The best average is also in this group. TanH best performing model with 2 layers produced interesting results for unseen data. It preformed overly well and adjusted to closer to the train set as time went on.

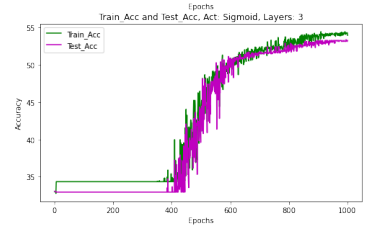


4.4.2 2 - 3 layer

2 and 3 layer models performed very similarly. very similar average and standard deviation. TanH and SiLU tends to be better on average here and Sigmoid has started to preform worse here(more in the explanation under Sigmoid). As can be seen in the graph below Sigmoid produced very aggressive changes compared to other smoother graphs like SoftPlus, which is showing consistent improvements as the layers increase.







4.4.3 4 - 5 layer

TanH and SiLU are very similar with TanH being slightly better. SoftPluse and ReLu are begining to drag behind compared to the other 2. And Here Sigmoid has a massive drop in accuracy.

4.4.4 Overall

Overall it can be seen that TanH and SiLU performed better with less more, ReLU and SoftPluse stay close to the same in Accuracy, where Sigmoid become very bad very fast with more layers due to the vanishing gradient issues it has.

4.5 Activation Function

4.5.1 ReLU and SoftPlus

ReLU and SoftPlus had very similar showings. had a very consistant showing regardless of number of layers. ReLu'S lower number at 4 layers could be due to variance of only doing 5 runs for an average. SoftPlus's decline at 5 might also be variance or it could be signs of the activation function also performing worse at higher layers.

4.5.2 Sigmoid

Due to Sigmoid's vanishing gradient issues, it had a significant drop accuaracy at 4+ layers. More Testing is needed to see if it would preform better with a large number of nodes in 1 to 3 layer.

4.5.3 TanH and SiLU

TanH preformed the best overall, with SiLU close behind.Both had low Standard Deviations but a noticeable increase in accuracy over layer increases. Possible being able to reach much better accuarcy at might number of layers and a better number of epochs to give more time to train.

5 Source Control

The Github repository can be found here:
https://github.com/David-Roodt/COS771-A2