LINEAR ALGEBRA *behind* *Artificial Neural Networks*

Sakshi Shetty, PES1201800190
Snigdha S Chenjeri, PES1201800045
Sruthy S, PES1201801143

1. INTRODUCTION

Our project revolves around the advanced applications of linear algebra. We have used data analytics to preprocess our dataset and derive inferences. Based on these inferences, we have built artificial neural networks to predict particular traits in animals.

2. DATASET

Our dataset contains of 101 animals and their various respective traits: physical traits such as aquatic/airborne/teeth etc and characteristic traits such as predator/venomous/domestic and class type. Upon analysing our data, our main focus is on the **4 traits** mentioned above.

3.1. DATA ANLYTICS

The first step was to clean the dataset by replacing missing and negative values with zero, and further replacing non numerical characters with zero. After this, we visualised the data to observe interdependencies of traits and their correlations, along with normalisation.

3.2. THE ARTIFICIAL NEURAL NETWORK - 4 MODELS

We chose 4 traits—venomous, predator, domestic & class type—as our data to predict. For each model, we used different activation functions—sigmoid, softmax and tanh—to compare the accuracy for each. Using the rest of the columns as the training data, a test data was made for the above three (one by one) and the accuracy measure determined how correctly our model evaluated the test data, i.e. how similar the test data and original data are.

4. LINEAR ALGEBRA USED

In the subsequent sections, we will describe in detail, according to our understanding, how linear algebra's core concepts are the foundation for the very existence of machine learning.

4.1. VECTORS

Every time we use a dataset, each of its **rows** are split into and represented as vectors—the combination of these vectors make the data matrix and result in our dataset as we see it.

4.2. LABELLING & ENCODING

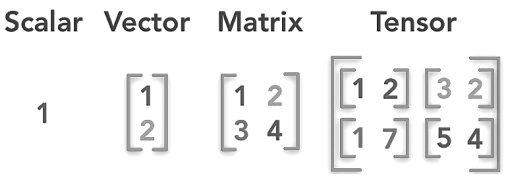
All categorical variables (strings, characters and the like) need to be converted to numerical format to be processed by a neural network. Each row, in such cases, is encoded as a **binary vector**, i.e. a vector with 0 or 1 value. For instance: in our dataset, if an animal is venomous, it is indicated by '1', and if it's not, it's indicated by '0'.

4.3. NORMALISATION

Normalisation is the process of taking data from a problem and reducing it to a **set of relations** while ensuring data integrity and eliminating data redundancy. Data integrity—all of the data in the database are consistent, and satisfy all integrity constraints.

4.4. TENSOR

The very core component of any neural network is tensors. To put it simply, tensors house data in N dimensions (matrices are hence 2D tensors). The popular python library tensorflow is aptly named so.



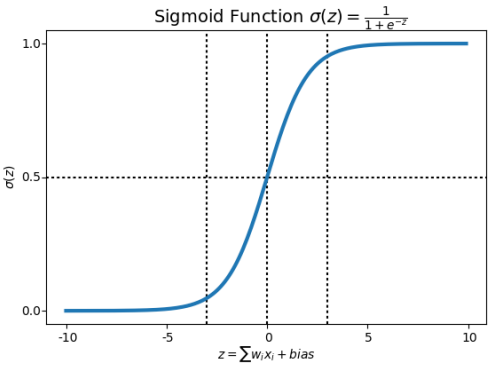
In ML, they encode **multi dimensional data**. For example, images are represented by 3 dimensions—height, width, depth (colour)—but while dealing with huge amounts of data, the sample size becomes the 4th dimension. Hence, 4D tensors can neatly pack images for further processing.

4.5. ACTIVATION FUNCTIONS

Neural network activation functions are a crucial component of deep learning. Activation functions determine the output of a deep learning model, its accuracy, and also the computational efficiency of training a model—which can make or break a large scale neural network. In the subsequent sections, we explain in detail the three functions we have used in our model - **sigmoid, softmax and tanh.**

4.5.1. SIGMOID

The sigmoid function is used because it exists between (0 to 1). Therefore, it is especially used for models where we have to predict the probability as an output. Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice. The formula and graph are as below:



It gives a smooth gradient, preventing "jumps" in output values. The output values are bound between 0 and 1, normalising the output of each neuron.

4.5.2. SOFTMAX

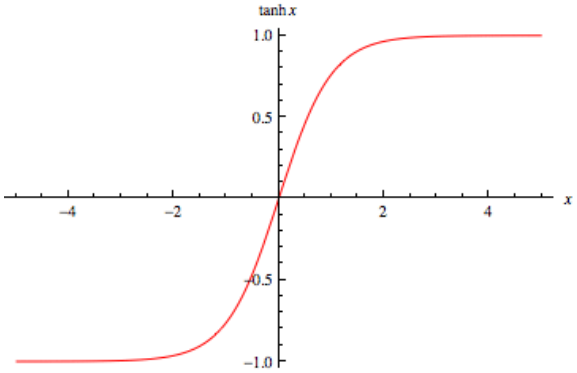
Softmax is often used in neural networks, to map the non-normalized output of a network to a probability distribution over predicted output classes.

$$\sigma(z)j = \frac{ezj}{\sum_{k=1}^{k} ezk} \quad for\ j = 1....k.$$

It can handle multiple classes—only one class in other activation functions normalizes the outputs for each class between 0 and 1, and divides by their sum, giving the probability of the input value being in a specific class. Softmax is useful for output neurons—typically Softmax is used only for the output layer, for neural networks that need to classify inputs into multiple categories.

4.5.3. TANH

The logistic sigmoid can cause a neural network to get "stuck" during training. This is due in part to the fact that if a strongly-negative input is provided to the logistic sigmoid, it outputs values very near zero. Since neural networks use the feed-forward activations to calculate parameter gradient. It makes it easier to model inputs that have strongly negative, neutral, and strongly positive values.



5. CONCLUSION

In conclusion, neural networks can only be built using activation functions, which are in turn constructed using tensors, matrices and vectors. All of these mathematical components are brought to life by utilising linear algebra. Therefore, the entirety of data science and machine learning relies heavily on linear algebra for its very existence.

REFERENCES

1. *"Neural Network Architectures and Learning" - Paper by Bogdan M Wilamowski*
2. *"Neural Network Design" - Book by Hagan, Demuth & Beale*
3. *"Machine Learning" - Book by Thomas G. Dietterich*
4. *"Searching for Activation Functions" - Paper by Ramachandran, Zoph & Le*