**Homework 4**

Neural Network

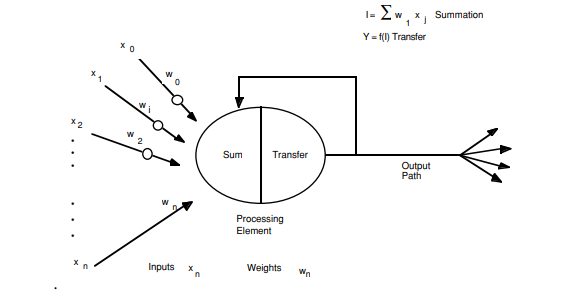
Shruti Avinash Pawar

June 20, 2020

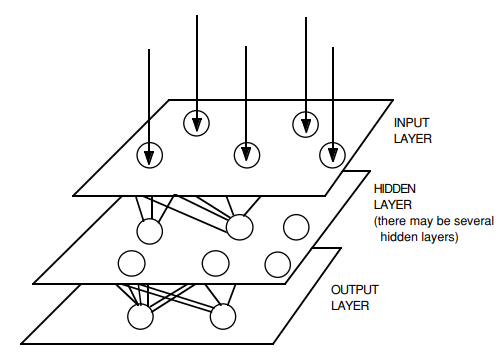
**Class:** ALY 6020 – Predictive Analytics

**Professor:** Dr. Marco Montes de Oca

Neural network is a deep learning algorithm that is based on the neural structure of the brain. In a human neural structure, a neuron acts as a control which conveys information via a host of pathways. The neural network algorithm replicates this aspect of the human brain and builds a neural network model using neurons. Below is a snapshot of the neural network model:



x0, x1,x2,…xn are inputs of the model. These inputs are assigned weights and transferred to a neuron. In the neuron two actions take place, first is the summation of inputs\*weights and second is the transfer where activation function is applied. (Anderson & McNeill, 1992) Activation function can be in the form of sigmoid, Rectified Linear Unit (Relu) or Tanh which is also known as Hyperbolic



In the picture on page 1, we can see three layers named as input layer, hidden layer and output layer. The hidden layers are a network of neurons and there can be several hidden layers in a neural network model (Anderson & McNeill, 1992).

**Activation functions:** In order to make the output of a neuron non-linear, activation function is used. Without an activation function the neural network model works same as the linear regression model. Neuron activations make the model capable of performing complex tasks and learning better.

1. Sigmoid Activation: F(x) = 1/(1+exp(-x), Function value ranges between 0 and 1.
2. Tanh Activation: tanh(x) = ex-e-x / ex+e-x ,Function value ranges between -1 and 1.
3. Relu Activation: A(x) = max(0,x) , gives an output x if x is positive or 0. Function value ranges between 0 and Infinity.

Sigmoid activations are used in the output layer because the function ranges between 0 and 1 which helps binary classification. Tanh is used in the hidden layer because the function ranges between -1 and 1 which brings the mean of the hidden layer close to 0. This function centers the data bringing the mean close to 0 and learning for the next layer becomes easier. Relu function works best in the hidden layers as the function ranges from 0 to infinity making the network sparse and efficient in learning (Omkar, 2019).

**Loss functions:**A loss function is a method of evaluating how well the algorithm models the dataset. Loss function will be higher if the predictions are not correct and lower if the predictions are good. This is because a loss function is the absolute difference between the predicted output and the actual output.

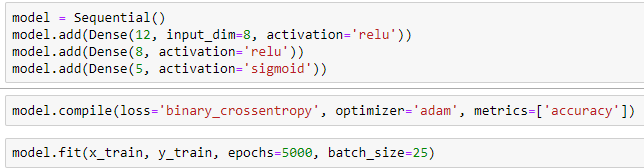
1. MSE: The loss function considers average of squared difference of errors.
2. Likelihood Loss: This function is a multiplication of predicted probabilities of each input.
3. Log loss or cross entropy: This function is a log version of the likelihood function. Formula for this function looks like: -(ylog(p) +(1-y)log(1-p))

When the actual class is 1, the second half of the cross entropy function will disappear whereas when the class is 0, the first half will disappear. ("Introduction to Loss Functions", 2018)

**Epochs, Batch Size and Iterations:** In general cases, neural network models are trained on a big data. It is difficult to pass the entire data in one go from the network thus the data needs to be cut down into batches. One epoch happens when the entire data passes from the neural network back and forth once. The dataset is split in batches and then multiple epochs are performed to train the model. Iterations is the number of times required to pass all batches through the network to complete one epoch. For example if the sample is 100 and batch size is 20, number of iterations required to complete one epoch is 5.(Sharma, 2017)

**Optimizers:** Deep learning neural networks are trained using gradient descent algorithms and learning rate is an important hyperparameter of the model. The step size of the weights updated during training the model is the learning rate. Smaller learning rates require more epochs and higher learning rates require lower epochs. Adam optimization algorithm is an extension to the gradient decent. Gradient decent maintains a single learning rate for all weights throughout the learning process. Adam algorithm calculates an exponential moving average of gradient and decays the learning rate per parameter. (Brownlee, 2017)

**Building a neural network:**To understand neural network models better, a dataset is picked which consist of two numbers encoded in a binary format with 4 bits. Thus, input dimension of the model is 8. The output variable is an addition of the two numbers encoded in a binary format with four bits and 1 carry. Thus, the output variable has 5 values in the array. There are 256 rows in the dataset. The tool used to perform this experiment is Jupyter Notebook Python and the libraries used are pandas, numpy and keras. The said data is imported in python using pandas dataframe. First 8 columns which are the binary inputs are converted into x matrix and the 5 output columns are converted into y matrix. The data is then split into train and test in the ratio of 75:25. The first neural network model is fit using the below code:



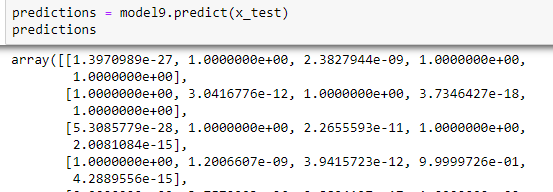
Let’s, understand the components of the code. In the first step, sequential() is used to add layers to the model. The first layer has 12 neurons with 8 input numbers and the activation function is relu. The number of neurons decrease in the hidden layer and the output layer has 5 output neurons as the output array has 5 values and the activation function used is sigmoid to predict binary class 0 or 1 effectively. Optimizer used in this code is adam. Epochs for this code is set at 5000 and batch size at 25. In the experiment, 10 different models were run using the above code with different activation functions, Epochs, batch sizes, number or neurons and number or layers. The loss function used throughout is binary cross entropy.

**Table 1:** Model summary of 10 neural network models

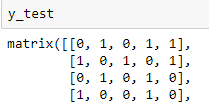
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Model Name** | **Layer 1 Neurons** | **Layer 2 Neurons** | **Layer 3 Neurons** | **Layer 4 Neurons** | **Optimizer** | **Activation** | **Epochs** | **Batch Size** | **Accuracy** |
| Model | 12 | 8 | 5 | - | adam | relu and sigmoid | 5000 | 25 | 85.63% |
| Model1 | 16 | 12 | 10 | 5 | adam | relu and sigmoid | 10000 | 25 | 99.37% |
| Model2 | 16 | 12 | 8 | 5 | LR = 0.01 | relu and sigmoid | 10000 | 25 | 89.69% |
| Model3 | 16 | 12 | 8 | 5 | LR = 0.05 | relu and sigmoid | 10000 | 25 | 53.75% |
| Model4 | 16 | 12 | 8 | 5 | LR = 0.01 | relu and sigmoid | 30000 | 25 | 96.88% |
| Model5 | 16 | 12 | 8 | 5 | adam | relu,tanh, and sigmoid | 10000 | 25 | 94.69% |
| Model6 | 16 | 12 | 8 | 5 | adam | relu and sigmoid | 15000 | 25 | 83.44% |
| Model7 | 16 | 12 | 8 | 5 | adam | relu and sigmoid | 10000 | 30 | 97.81% |
| Model8 | 24 | 16 | 10 | 5 | adam | relu and sigmoid | 10000 | 30 | 99.06% |
| Model9 | 30 | 20 | 10 | 5 | adam | relu and sigmoid | 10000 | 30 | 100% |

Referring to table 1, we can see that the first model has only 3 layers with lower epochs and accuracy of 85.63%. To increase the accuracy, another layer is added to the architecture and the number of neurons is also increased. The Epochs were increased to 10000. This has considerably increased the accuracy to 99.37%. Now, the target is to take the accuracy to 100%. Thus, to check if change in the optimizer results into an increase in the accuracy, in model2, learning rate 0.01 was used as an optimizer instead of adam. This decreased the accuracy to 89.69%. To check how a higher learning rate impacts the model, model3 was run with just change in learning rate to 0.05. This drastically decreased the model accuracy to 53.75%. This implies that lower learning rate gives a better accuracy. As higher epocs are required for a low learning rate, in model4, epochs were increased to 30000. This increased the accuracy to 96.88%. Another increase in epocs will result into an accuracy of 100% but that will increase the learning time or code runtime. Therefore, it was decided to continue with adam optimizer and tweak other aspects of the model. In model5, tanh was used as activation in the hidden layers, but it did not result into a higher accuracy. In model6, epochs were increased to 15000 but that is not sufficient to meet the 100% target. Keeping the epochs at 10000, batch size was increased to 30 in model7. This improved the accuracy from 83.44% to 97.81%. Now the optimizer is set at adam, batch size at 25, epochs at 10000 and activation at relu for hidden layers and sigmoid for final layers. Now, neurons are changed to check the impact on accuracy. In model8, neurons were increased in the first 3 layers keeping the other aspects fixed as discussed above. This resulted in an accuracy of 99.06%. Further increasing the neurons resulted in 100% accuracy. More number of neurons make the model more complex and efficient.

The prediction output of model9 looks as below:



Actual values of outcome are as follows:



The values that have eraised to minus sign correspond to 0 and those that have e raised to plus sign correspond to 1 in the predicted output. If we compare these values with the actual output, they match. Thus the 100% accuracy result is correct.

**Conclusion:**

To summarize, optimizing a neural network model requires the following steps:

1. Changing the architecture of the model which involves hidden layers
2. Deciding the loss function
3. Deciding the number of neurons
4. Deciding the activation function for the neurons
5. Optimizer selection
6. Choosing the optimum epochs and batch size for the model which do not increase the learning time

In this experiment trial and error was used to determine all the above aspects of the model in order to achieve accuracy of 100%.

**Bibliography:** Anderson, D., & McNeill, G. (1992). Artificial Neural Network Technology - A DACS State-of-the-Art Report. Retrieved 21 June 2020, from <http://andrei.clubcisco.ro/cursuri/f/f-sym/5master/aac-nnga/AI_neural_nets.pdf>

Omkar, N. (2019). Activation Functions with Derivative and Python code: Sigmoid vs Tanh Vs Relu. Retrieved 21 June 2020, from <https://medium.com/@omkar.nallagoni/activation-functions-with-derivative-and-python-code-sigmoid-vs-tanh-vs-relu-44d23915c1f4>

Brownlee, J. (2019). Your First Deep Learning Project in Python with Keras Step-By-Step. Retrieved 21 June 2020, from <https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/>

Introduction to Loss Functions. (2018). Retrieved 21 June 2020, from <https://algorithmia.com/blog/introduction-to-loss-functions>

Sharma, S. (2017). Epoch vs Batch Size vs Iterations. Retrieved 21 June 2020, from <https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9>

Brownlee, J. (2017). Gentle Introduction to the Adam Optimization Algorithm for Deep Learning. Retrieved 21 June 2020, from <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/#:~:text=What%20is%20the%20Adam%20optimization,iterative%20based%20in%20training%20data>.