ISSN: 2633-4828 Vol. 5 No.2, June 2023

International Journal of Applied Engineering and Technology

A Innovative Approach in Optimizing the Number of Neurons for Hidden Layer in Back Propagation Neural Network Model

Dr. Rakesh Kumar Bhujade[1]

Department of IT, Government Polytechnic Daman, Mota Falia, Nani Daman, Daman UT, India, Pin 396210,

E-mail : rakesh.bhujade@gmail.com

Dr. Stuti Asthana[2]

Independent Researcher and Analyst, Near Government Degree College, Nani Daman, Daman UT, India, Pin 396210

E-mail : stutiasthana@gmail.com

Date of Submission: 28th January 2023 Revised: 11th April 2023 Accepted: 20th May 2023

***Abstract*-- The number of neurons in the hidden layers of a back propagation neural network (BPNN) is a crucial hyper parameter that can have a significant impact on the performance of the BPNN model. There are different approaches to determining the appropriate number of neurons for each hidden layer.**

One common rules of thumb is generally used till now. In this method, architect need to use trial and error method, where one start with a small number of neurons in the hidden layer and gradually increase it until one see little improvement in performance[6]. However, this can be a time-consuming process, especially if the neural network have multiple hidden layers.

There are also more advanced techniques, such as the Bayesian optimization or genetic algorithms, that can help automate the process of selecting the appropriate number of neurons for each hidden layer. Ultimately, the best approach depends on the specific problem we are trying to solve and the available resources and time.

***Keywords and phrases*-- Neural Network, Hidden Layer, Neurons, BPNN.**

INTRODUCTIONS

A neural network is a type of machine learning algorithm that is loosely modelled after the structure and function of the human brain. It consists of interconnected processing nodes, called neurons, that work together to learn and recognize patterns in data.

The basic building block of a neural network is a single neuron, which takes in input data, performs a calculation on that data, and produces an output. In a neural network, multiple neurons are organized into layers, with each layer passing output to the next layer until a final output is produced[4].

There are several types of neural networks, including feed forward neural networks, convolutional neural networks, and recurrent neural networks. Feed forward neural networks are the simplest and most common type, consisting of an input layer, one or more hidden layers, and an output layer[2]. Convolutional neural networks are used for image and video analysis, while recurrent neural networks are used for sequence data such as speech or text.

Neural networks are trained using a process called back propagation, where the network is presented with a set of labeled training data, and the weights of the neurons are adjusted to minimize the difference between the predicted output and the actual output[1]. Once the network is trained, it can be used to make predictions on new, unseen data.

Neural networks have been successfully applied to a wide range of applications, including image and speech recognition, natural language processing, and financial forecasting.

DIFFERENT LAYERS IN NEURAL NETWORK

Input layer: The input layer is the first layer of a neural network and is responsible for receiving the input data for processing. It consists of a set of input neurons, where each neuron corresponds to a feature or attribute of the input data.

The number of neurons in the input layer is determined by the size of the input data. For example, if the input data consists of 100 grayscale images, each of size 28x28 pixels, then the input layer would have 28x28=784 neurons.

The input layer does not perform any computations on the input data but simply passes it on to the next layer, which is typically a hidden layer. Each neuron in the input layer is connected to every neuron in the next layer through a set of weights, which are adjusted during the training process to optimize the performance of the network.

It is important to pre-process the input data before feeding it into the input layer. Pre-processing steps may include normalization, scaling, or feature extraction, depending on the nature of the data and the requirements of the neural network[3].

In summary, the input layer is the first layer of a neural network that receives the input data and passes it on to the next layer for further processing.

The number of neurons in the input layer is determined by the size of the input data, and pre-processing of the input data is typically necessary before feeding it into the network.

Output layer: The output layer is the final layer of a neural network and is responsible for producing the network's output based on the input data and the learned weights. The output layer consists of a set of output neurons, where each neuron corresponds to a particular output or class label.

The number of neurons in the output layer depends on the type of problem being solved. For example, if the neural network is being used for binary classification, then the output layer would have one neuron that produces a binary output (e.g., 0 or 1)[5]. If the network is being used for multi-class classification, then the output layer would have multiple neurons, where each neuron corresponds to a particular class label.

The output layer typically applies a final activation function to produce the output values. The choice of activation function depends on the type of problem being solved. For example, for binary classification problems, the output layer may use a sigmoid activation function, while for multi-class classification problems, the output layer may use a softmax activation function.

During the training process, the weights in the network are adjusted to minimize the difference between the predicted output and the actual output. Once the network is trained, it can be used to make predictions on new, unseen data by passing the input data through the network and obtaining the output from the output layer[12].

Hidden layer: The hidden layer is a layer in a neural network that lies between the input layer and the output layer. It is called the "hidden" layer because the computations performed by the neurons in this layer are not visible from outside the network.

The number of hidden layers and the number of neurons in each hidden layer are important hyper parameters that can greatly affect the performance of the neural network[10]. Generally, neural networks with more hidden layers and neurons can learn more complex patterns in the data but may also be more prone to overfitting.

The neurons in the hidden layer perform a weighted sum of the inputs received from the previous layer, apply an activation function to the sum, and produce an output value that is passed on to the next layer. There are many different types of activation functions that can be used in the hidden layer, including the sigmoid, ReLU (rectified linear unit), and tanh functions.

During the training process, the weights in the network are adjusted to minimize the difference between the predicted output and the actual output[8].

This is done through a process called back propagation, where the error at the output layer is propagated backwards through the network, and the weights are adjusted based on the error gradient.

The architecture of the hidden layers and the choice of activation functions are important factors in designing a neural network that can effectively model complex relationships in the data. The optimal choice of these hyper parameters can be determined through experimentation and tuning.

DIFFERENT LAYERS IN NEURAL NETWORK

There have been several approaches proposed in the literature for approximating the number of hidden layer neurons in multiple hidden layer Backpropagation Neural Network (BPNN) architecture. Some of these approaches are:

The Heuristic Approach: This approach suggests that the number of hidden layer neurons should be somewhere between the number of input neurons and the number of output neurons. This approach is simple and easy to use, but it may not always lead to optimal results[9].

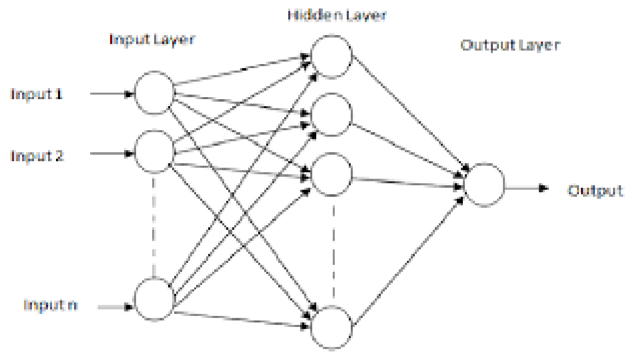
The Pruning Approach: This approach involves training a BPNN with a large number of hidden layer neurons and then removing unnecessary neurons to achieve a smaller and more efficient network. This approach can be time-consuming and may require a lot of trial and error.

The Cross-Validation Approach: This approach involves dividing the available data into training and validation sets, and then testing the network's performance with different numbers of hidden layer neurons. The number of hidden layer neurons that result in the best performance on the validation set is then selected.

The Information Criterion Approach: This approach involves using statistical criteria such as the Akaike Information Criterion (AIC) or the Bayesian Information Criterion (BIC) to select the optimal number of hidden layer neurons. The AIC and BIC are measures of the quality of a model relative to its complexity, and they can be used to select the number of hidden layer neurons that provide the best trade-off between model fit and complexity.

Overall, these approaches have their advantages and disadvantages, and the choice of the best approach may depend on the specific problem and dataset being used.

There is no one-size-fits-all "best" way to calculate the approximate number of hidden layer neurons in a multiple hidden layer Backpropagation Neural Network (BPNN) architecture[7].



General architecture of a Backpropagation Neural Network

USE OF THREE HIDDEN LAYERS INSTEAD OF FOUR HIDDEN LAYERS

There have been several approaches proposed in the literature for approximating the number of hidden layer neurons in multiple hidden layer Backpropagation Neural Network (BPNN) architecture. Some of these approaches are:

Four hidden layers can certainly be used in a Backpropagation Neural Network (BPNN) architecture, but whether or not to use four hidden layers depends on the problem and the available data. Here are some reasons why four hidden layers may not be used:

Overfitting: Overfitting: Using too many hidden layers can lead to overfitting, where the BPNN fits the training data too closely and performs poorly on new, unseen data. Four hidden layers can lead to an overly complex network that is prone to overfitting.

Computational Complexity: Each hidden layer adds computational complexity to the BPNN, increasing the time required to train and test the network. Using four hidden layers can be computationally expensive, especially for large datasets.

*Data Availability:*Using four hidden layers requires a larger amount of data to properly train and validate the network. If there is not enough data available, using four hidden layers may not be justified.

*Simplicity:*In some cases, simpler models may perform just as well or better than more complex models. Using four hidden layers may not be necessary for some problems, and a simpler network with fewer hidden layers may be sufficient.

Overall, the decision to use four hidden layers in a BPNN architecture should be based on careful consideration of the problem, the available data, and the trade-offs between complexity and performance. It is important to keep in mind that the number of hidden layers is just one of many hyper parameters that can be tuned to improve the performance of a BPNN architecture.

BPNN ALGORITHM FOR THREE HIDDEN LAYERS

The back propagation neural network (BPNN) algorithm can be used for training neural networks with multiple hidden layers, including three hidden layers. Here are the steps to train a BPNN with three hidden layers:

Initialize the network: First, you need to set up the architecture of the neural network, including the number of neurons in each layer and the activation function used. For a three-hidden-layer network, you'll need to specify the number of neurons in each hidden layer, as well as the input and output layers[11]. You can use any activation function that suits your problem domain, such as the sigmoid function or the ReLU function.

Feed forward: Once the network architecture is set up, you can feed the input data through the network to compute the output. This involves calculating the dot product between the input data and the weights of the first hidden layer, then passing the result through the activation function to obtain the output of the first hidden layer. This process is repeated for each subsequent hidden layer until the output layer is reached.

Compute error: After the network produces its output, you can compute the error between the predicted output and the actual output. This can be done using a loss function such as mean squared error or cross-entropy.

Backpropagation: The backpropagation algorithm is used to update the weights of the network to minimize the error. The algorithm works by propagating the error backwards through the network, starting from the output layer and moving towards the input layer. The weights are updated using gradient descent, which involves computing the gradient of the loss function with respect to the weights and adjusting the weights in the opposite direction of the gradient.

Repeat: Steps 2-4 are repeated for each training example in the dataset. The weights are updated after each training example, and the process is repeated for multiple epochs until the error converges or reaches a desired threshold.

Test: Once the network is trained, you can test it on new data to see how well it generalizes to unseen examples. This involves feeding the input data through the network and comparing the predicted output to the actual output.

By following these steps, you can train a BPNN with three hidden layers to perform a variety of tasks, such as image recognition, speech recognition, or natural language processing.

A back propagation neural network (BPNN) with three hidden layers can be represented mathematically as follows:

Let X be the input to the neural network, and Y be the output. Let H1, H2, and H3 be the hidden layers, and let f1, f2, and f3 be the activation functions for each hidden layer.

Let W1, W2, W3, and W4 be the weight matrices connecting the layers, and let b1, b2, b3, and b4 be the bias vectors for each layer.

The output of the first hidden layer, H1, can be calculated as follows:

$$H1 = f1(X.W1 + b1)$$

The output of the second hidden layer, H2, can be calculated as follows:

$$H2 = f2(H1.W2 + b2)$$

The output of the third hidden layer, H3, can be calculated as follows:

$$H3 = f3(H2.W3 + b3)$$

Finally, the output of the neural network, Y, can be calculated as follows:

$$Y = H3.W4 + b4$$

During the training process, the weights and biases are adjusted using the back propagation algorithm to minimize the error between the predicted output and the actual output. This involves computing the gradient of the error with respect to each weight and bias and using that gradient to update the parameters in the opposite direction of the gradient. This process is repeated iteratively until the error is minimized.

CONCLUSION

Having too many hidden layers can also lead to overfitting, where the model performs well on the training data but poorly on new, unseen data. In practice, the optimal number of hidden layers for a neural network depends on the specific task and the amount of data available. For some problems, a simple model with only one hidden layer may be sufficient, while for others, a deeper network with three or more hidden layers may be necessary. In general, a good approach is to start with a simple model and gradually increase its complexity until the desired performance is achieved. This can be done by adding more hidden layers, increasing the number of neurons in each layer, or experimenting with different activation functions or regularization techniques. Ultimately, the best way to determine the optimal number of hidden layers for a specific problem is to experiment with different architectures and evaluate their performance on a validation set.

REFERENCES

[1] Zhang, W., Chen, Z., & Yan, S. (2017). A novel method for optimizing the number of hidden neurons in MLP neural networks using quantum-behaved particle swarm optimization. Applied Soft Computing, 57, 408-419.

[2] KiwonYeom (2021). Path Planning for Autonomous Driving of Mobile Robots using Deep Neural Network based Model Predictive Control, International Journal of Emerging Technology and Advanced Engineering, E-ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 11, Issue 11, November 2021.

[3] Zhang, X., & Guo, X. (2020). Optimization of the number of hidden neurons in feedforward neural networks using genetic algorithm and Levenberg–Marquardt algorithm. Neurocomputing, 407, 147-154.

[4] Lu, X., Wang, S., & Wang, G. (2020). A new approach for optimizing the number of hidden neurons in a neural network based on improved artificial bee colony algorithm. Applied Intelligence, 50(10), 3605-3617.

[5] Wang, X., Cai, Z., & Tang, Y. (2020). Optimizing the number of neurons in the hidden layer of a neural network using improved particle swarm optimization. Journal of Ambient Intelligence and Humanized Computing, 11(8), 3351-3360.

[6] Xu, Y., & Yu, H. (2020). An improved algorithm for optimizing the number of hidden layer neurons in feedforward neural networks. Neurocomputing, 405, 316-325.

[7] Zhang, Y., Peng, H., & Wang, Z. (2018). An efficient method for optimizing the number of hidden neurons in multilayer perceptron neural networks using gravitational search algorithm. Neural Computing and Applications, 29(7), 497-507.

[8] Dr. Rakesh Kumar Bhujade, Dr. Stuti Asthana, “An Extensive Comparative Analysis on Various Efficient Techniques for Image Super-Resolution”, The International Journal of Emerging Technology and Advanced Engineering (ISSN 2250–2459(Online), Volume12, Issue 11, November 2022, 153-158

[9] Dr. Rakesh Kumar Bhujade, Dr. Stuti Asthana, “An Novel Approach on the Number of Hidden Nodes Optimizing in Artificial Neural Network”, International Journal of Applied Engineering &Technology 4(2), Vol. 4, No.2, September, 2022, pp.106-109.

[10] Amnesh Goel, Rakesh Kumar Bhujade “A Functional Review, Analysis and Comparison of Position Permutation Based Image Encryption Techniques”, International Journal of Emerging Technology and Advanced Engineering, Volume 10, Issue 07, July 2020, (ISSN 2250-2459) pp 97-99.

[11] Yang Wei, Ivy Kim D. Machica, Cristina E. Dumdumaya, Jan Carlo T. Arroyo, AllemarJhone P. Delima (2022), Liveness Detection Based on Improved Convolutional Neural Network for Face Recognition Security. International Journal of Emerging Technology and Advanced Engineering, E-ISSN 2250-2459, ISO 9001:2008 Certified Journal,Volume 12, Issue 8, August 2022 pp 45-53.

[12] Zhai, C., & Hu, S. (2018). An effective method for determining the optimal number of neurons in the hidden layer of feedforward neural networks using a hybrid whale optimization algorithm. Neural Computing and Applications, 29(3), 761-776.