CHAPTER – 8
Optimization of BPNN (OBPNN)

8.1 Introduction
Chapter 3 is focused on Back Propagation Neural Network (BPNN) and its parameters. This chapter addresses the need for optimization of parameters and a model to optimize them. To achieve the objective, we surveyed various literatures and noted various possible methods to set values of each parameter. As per the optimization model discussed in this chapter, each of these possible methods had been implemented on 3 Variable XOR problem and few best methods for each parameter had been selected. For selection of the best method, each of these best methods was tested on KDD CUP 1999 dataset. By comparing the results of various best methods, optimum value for each parameter were selected. More detail about the optimization model is available in our previous work [1].

8.2 Need for Parameter Optimization
During literature review and implementation, we observed that BPNN suffers from challenges like: high training time, high complexity and local minima. One of the approaches to resolve these challenges is by optimizing the BPNN’s parameters / criteria with respect to the dataset which is supposed to be used for training and testing the model. In the first phase of the experiments, our research model uses KDD CUP 1999 dataset. These dataset has very large number of records with each record having 41 features. Such high number of features increases the complexity of BPNN. As a result, it is very difficult to optimize the parameters/ criteria of BPNN. To do so, we present a methodology in this chapter. By using this methodology, we were able to optimize the parameters / criteria of BPNN. Such Optimized BPNN (OBPNN) reduces the response time and complexity and overcomes from local minima.

8.3 Parameters of BPNN
From our literature review and also from our experiments, we found various parameters/ criteria that affect performance of the BPNN. Following is the list of such parameters / criteria [1].
8.4 The Model for Parameter Optimization

To optimize the parameters / criteria of BPNN, we proposed following model. More details of the optimization model are available in our previous work [1].

Step 1: By referring the literatures, for each parameter / criteria, collect various approaches to assign the values.

Step 2: For each parameter / criteria, implement all these approaches on any small BPNN model and study the effect of these parameters / criteria.

Step 3: On the basis of the above study, decide few good approaches or values for each parameter / criteria.

Step 4: For each parameter / criteria, implement BPNN with these good approaches on limited records of large dataset. Compare the results and find the best approach or optimal value for each parameter / criteria.

8.5 Implementation

We implemented the parameter optimization model and optimized parameters / criteria of BPNN. For the implementation, we used NSL dataset which is modified version of KDD CUP 1999 Dataset. Following is the details of such implementation.

8.5.1 Effect of Learning Rate

To assign the learning rate, various approaches which includes: fixed learning rate and variable learning rate are available. Further, for the case of fix learning rate, various possible values can be: 0.01, 0.02, 0.03,...... , 0.99. To observe the effect of the fix learning rate on 3 Variable XOR BPNN, we did our implementation with learning rate ranging from 0.1 to 0.9 increasing in step of 0.1. We took other parameters as constant. Fig.8.1 shows the effect of the learning rate on 3 Variable XOR BPNN implementation. As per the Fig.8.1, by increasing the learning rate, time taken to learn the given problem...
gradually decreases. We achieved best learning rate near to 1.

![Graph showing the mean learning time vs. learning rate](image)

**FIGURE 8.1:**

**Effect of Learning Rate in 3 Variable XOR Problem.**

**Learning Time is Minimum When Learning Rate Approaches 1.**

One more approach to assign the learning rate is variable learning rate which is known as dynamic assignment approach. Comparison of fixed learning rate with variable learning rate is shown in Table 8.1. For this comparison, we used BPNN with NSL Data Set. As compared to 0.1 learning rate, learning rate assigned by method of [7] and [12], reduces the number of the epochs by 81% and 90% respectively. As per the Table 8.1, DALRM method of [12] looks more promising for the anomaly detection using NSL dataset.

**TABLE 8.1:**

<table>
<thead>
<tr>
<th>Learning Rate</th>
<th>Number of Epochs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>35237</td>
</tr>
<tr>
<td>0.9</td>
<td>20245</td>
</tr>
<tr>
<td>Dynamic Change Learning Rate Strategy [7]</td>
<td>6697</td>
</tr>
<tr>
<td>DALRM [12]</td>
<td>3523</td>
</tr>
</tbody>
</table>

Our experiments on BPNN show that high learning rate takes less training time, but learning is not matured. To predict the new and modified attack with good confidence, learning must be matured. To support the mature learning, learning rate must be as low as
possible. On the other side, in the case of the anomaly detection, where the model has to learn very large number of the records, low learning rate is not practically accepted as it takes very long time to learn. So, to support the mature and fast learning, variable learning rate strategy suggested by authors of [7] and [12] should be used.

8.5.2 Initial Weight
Weights in the BPNN can be set by a fixed value or variable values. To observe the effect of fixed weight, we performed various experiments with fixed initial weight ranging from 0.1 to 0.9. We performed experiments on 3 Variable XOR Problem by taking other parameters as constant. Results of the experiments are shown in Fig.8.2. As per the Fig.8.2, if we set the initial weights near to 0, learning will be very slow, which gradually becomes faster as we increase the initial weight. At weights near to 0.5, learning is very fast. After that, as we increase the initial weights, the learning time increases. It should be noted that for the 3 Variable XOR Problem, arithmetic mean of the outputs is 0.5, and for 0.5 initial weights, we got the lowest learning time.

![Figure 8.2: Effect of Fixed Weights in 3 Variable XOR Problem. Learning Time is Minimum When Initial Weights are Near to 0.5.](image)

One more approach for initializing the weights is by random values. Authors of [6] [7] and [10] had discussed such approaches. We did 100 experiments on 3 Variable XOR Problem, with initializing the weights by random values, and taking other parameters as constant. From our experiments, we observed that random initial weights decrease the learning speed but learning is more matured. During these experiments, we got 0.92 seconds as mean learning time which is 11% higher than lowest learning time, achieved by setting the initial
weights at arithmetic mean of the output values.

To be more precise, we performed experiments on 10000 rows of NSL dataset by setting the initial weights at arithmetic mean of the output values and random values, and recorded the results for the first 200 epochs. For each case, we recorded the number of complex rows, the row for which more than one iterations are required for learning. Table 8.2 shows the results of our implementation. As per the Table 8.2, total number of complex rows is less by 23% for the case of initial weights at arithmetic mean value of outputs as compared to the random initial weighs. We got similar results for 3 Variable XOR BPNN also. So, for training BPNN using NSL dataset, our experiments suggested to initialize the weight at mean of outputs instead of random weights.

**TABLE 8.2:**

<table>
<thead>
<tr>
<th>Initial Weights</th>
<th>Total Number of Complex rows Processed for first 500 epochs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Weights</td>
<td>110862</td>
</tr>
<tr>
<td>Weights at Arithmetic Mean of the Output Values (0.5)</td>
<td>90080</td>
</tr>
</tbody>
</table>

One more approach to set the initial weights is described in [3] and available in (8.1).

\[
\text{Initial Weight} = [-\text{Epsilon}, \text{Epsilon}] \hspace{1cm} \text{...} \hspace{1cm} (8.1)
\]

Where \(\text{Epsilon} = \frac{\sqrt{6}}{\sqrt{\text{previous layer neurons} + \text{next layer neurons}}}\)

To compare this approach with approach of initializing weights at mean value of outputs, we performed 100 experiments. Results of our experiments shows that BPNN with initial weights assigned by (8.1), outer performs on mean values, by many folds.

**8.5.3 Number of Hidden Units**

Number of the hidden units in BPNN decides the complexity of the system. To reduce the complexity, if few units are used then it leads to high training time and response time. On the contrary, the system with too many hidden units, increases the complexity which in turn leads to high training time and response time [8] [13]. Thus, for selection of number of hidden units, topmost care should be taken.
To check the effect of number of hidden units, we implemented BPNN for 3 Variable XOR Problem. We took BPNN with single hidden layer and varied number of hidden units ranging from 1 to 20. During these experiments, we took other parameters as same. Results of our experiments are shown in Fig.8.3. As shown in Fig.8.3, initially when number of the hidden units were low, training time was very high. This training time gradually decreases as we increased the number of hidden units. At 7 hidden units, training time was minimum, which shows the optimal number of hidden units for 3 Variable XOR Problem. After that, adding more hidden units increased the training time. When we extended our experiment up to 50 hidden units, we observed similar behavior. After 28 units, training time increases rapidly and becomes 3.4 seconds at 50 hidden units, which was 0.75 seconds for 7 hidden units.

To decide the exact number of the hidden units, researchers proposed various formulas. Authors of [14] had suggested \( (8.2) \) while authors of [2] had suggested \( (8.3) \) and \( (8.4) \) formulas.

Number of Hidden Units = \( 2n+1 \) ......................... \( (8.2) \)

Where \( n \) is number of the inputs

\[
Total \ Units = \sqrt[2]{(m + 2)^N} \quad (<< N)
\]

---

**FIGURE 8.3:**

Effect of Hidden Units on 3 Variable XOR BPNN. Initially, by Increasing Number of Hidden Units, Learning Time Decreases. At 7 Hidden Units, the Learning Time is Minimum.
Implementation

\[ L1 = \sqrt[2]{(m + 2)^N} + 2 \frac{\sqrt{N}}{m + 2} \] .......................... (8.3)

\[ L2 = m \frac{\sqrt{N}}{m + 2} \] .......................... (8.4)

Where \( m \) = number of output units and \( N \) = Number of input units.

In case of 3 Variable XOR Problem, according to (8.2), 7 hidden units are required. Similarly, as per (8.3) and (8.4), total 6 hidden units out of which 5 in layer 1 and 1 is in layer 2 are required. As per our experiment also, optimal number of the hidden units is 7. So, our experimental results on 3 Variable XOR Problem are very close with [2] and [14].

But, for the case of anomaly detection using KDD CUP or NSL Dataset which has 41 inputs and 1 output, as per [14], 83 hidden units are required, while as per [2], 22 hidden units are required. There is a very huge difference of 61 hidden units in both these approaches. The question arise is which one is the best? To answer the question, we trained BPNN by first 10000 rows of NSL Data set. During the experiments, we varied the number of hidden units at layer 1 and layer 2. Table 8.3 shows the results of the same experiments. From the Table 8.3, it can be seen that BPNN with 18 units in hidden layer1, and 4 units in hidden layer 2 takes 98% less time as compared to the BPNN with 82 and 1 hidden units in layer 1 and layer 2 respectively. So, method presented in (8.3) looks more promising for the case of anomaly detection using KDD CUP or NSL dataset.

<table>
<thead>
<tr>
<th>Number of Hidden Units in Layer 1</th>
<th>Number of Hidden Units in Layer2</th>
<th>Total Training Time for 200 Epochs (In Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>82 Units</td>
<td>1 Unit</td>
<td>2711.719</td>
</tr>
<tr>
<td>41 Units</td>
<td>41 Units</td>
<td>1792.875</td>
</tr>
<tr>
<td>18 Units</td>
<td>4 Units</td>
<td>55.216</td>
</tr>
</tbody>
</table>

8.5.4 Overtraining and Early Stopping

As per [5], during the training, weights moves away from initial values towards better values and decreases the training error. At some point of time during training, all the weights set to their best values which lead to minimum error for all the learning samples.
Still if BPNN is over trained then it increases the complexity and leads to poor generalization. Hence, learning must be stopped early to eliminate this problem. Early stopping criteria helps to overcome from the overtraining. During our implementation of 3 Variable XOR Problem, we observed the following criteria which can be used for early stopping when:

1) Error is less as compare to the expected error.
2) Error does not decrease further.
3) Weights do not change much from one iterations to the next couple of iteration.
4) Number of the epochs crosses the maximum level.
5) More iteration has been done for the given input rows.

To set the early stopping points for the BPNN using NSL dataset, we used all the above criteria in our implementation. When we compared the implementation of BPNN with all above early stopping criteria (Case 1) with one or two early stopping criteria (Case 2), we observed that for the case 1 training time was reduced by a large extent as compare to the case 2. Due to the early stopping criteria (case 1), only 0.15% of the input pairs were not learned properly. So, from our experiments and observation, we suggest that for the case of training the BPNN for anomaly detection, all the above early stopping criteria should be used.

8.5.5 Learning Samples

During training the BPNN, fewer samples reduce the detection rate, while more samples increase the complexity and training time [4]. For optimal number of samples, authors of [4] suggested (8.5) formula.

\[
\text{# weights} = h \times (s+1) + h + 1 \quad \ldots \quad (8.5)
\]

Where \( s \) is inputs, \( 1 \) is output and \( h \) is hidden node.

As per the authors, for each weight, minimum 30 samples should be taken. For the better results authors suggested to take 100 samples for each weight. For example: 15 inputs, 1 output and 10 hidden node implies 171 weights and 17100 samples.

In the case of anomaly detection, KDD CUP 1999 and NSL Data set are commonly used. KDD CUP 1999 dataset, have 494021 training records, while NSL dataset have 125973 training records [9] [11]. These both the dataset has 41 inputs and 1 output. To train the BPNN with these datasets, as per [4], 96700 samples are sufficient. Hence, both KDD CUP 1999 and NSL are best dataset as per number of the samples. However, during our various
implementations, we found that the model which is trained from KDD CUP 1999 dataset gives better accuracy as compared to NSL dataset. Hence, we suggested to use KDD CUP 1999 dataset in the proposed model.

8.5.6 Activation Function

Any function which is continuous and differentiable is a candidate for the activation function for BPNN [10]. As per [4] and [10], there are various types of activation functions which are applicable to the neural network. Followings are those activation functions.

1) Identity Function
2) Binary Step Function
3) Bipolar Step Function
4) Logistic Sigmoid Function
5) Hyperbolic Tangent Function
6) Ramp Function

Performance of the above activation functions are varying from case to case. For the case of the 3 Variable XOR problems, when we compared the results of the tangent sigmoid function with logistic sigmoid function, we found that, tangent sigmoid function is better. For case of anomaly detection, majority of the authors used logistic sigmoid function [1]. When we compared the BPNN with logistic sigmoid and hyperbolic tangent function, we found that the processing of hyperbolic tangent function is faster as compared to the sigmoid activation function. Number of the complex rows was very less for sigmoid activation function as compared to hyperbolic tangent function. On the basis of the number of the complex rows left during training, sigmoid is a better choice for training BPNN using KDD CUP 1999 dataset.

8.5.7 Normalization of Inputs

As per [4], inputs of BPNN must be in a normalized form. BPNN can performance better with the small values of the inputs, typically around [-1, 1]. In case of the 3 Variable XOR problems, we do not have inputs other than zero and one. But in case of anomaly detection by KDD CUP 1999 dataset, we have 41 inputs and 1 output. These attributes are not in normalized form. To normalize them, transformation, cleaning, scaling, and re-sequencing must be done. When we used normalized KDD CUP 1999 dataset to train the BPNN, we found the training time is reduced by a large extent.
8.6 References

3. Andrew NG, Machine Learning Course Exercise -4, Coursera, Stanford University.
9. Faculty of Computer Science University of New Brunswick, NSL- KDD Dataset, , http://www.iscx.ca/NSL-KDD.