Analysis of Genetic Algorithms and Distinctiveness in Back Propagation Neural Network on Diabetic Retinopathy classification

Abu Abraham,

Research School of Computer Science,
The Australian National University, Canberra, Australia, 2601.

Abstract. Neural Networks are of growing interest in the fields of pattern recognition and regression. In this report, the performance of a Back-propagation neural network model is assessed based on its classification accuracy on the Diabetic Retinopathy Debrecen dataset. The efficiency of the implemented model was observed to be similar to other neural network models. In neural networks, determining the hyper-parameters is a tough problem to solve. This paper addresses a solution to that problem by implementing a genetic algorithm to find the best hyper-parameters. Moreover, a pruning technique, distinctiveness, is applied to further reduce the hidden units. Various other performance improvement techniques such as mini-batch training are also explored.

Keywords: Back-propagation neural-network, Genetic Algorithm, Distinctiveness

1 Introduction

Neural networks have attracted tremendous interests from various fields of study [5]. The use of neural networks in the field of medicine is growing. The ability of such models to understand and handle large number of features makes it an attractive option over the traditional statistical methods [4].

In this paper, an analysis of the back-propagation neural network with various state of the art approaches are studied using the Diabetic Retinopathy Debrecen dataset. The Retinopathy dataset was an attractive choice, aligning with the global trend of using artificial neural networks in medical data [6]. Moreover, being a binary classification dataset, it was an ideal choice to showcase various approaches mentioned further in this report.

The pre-processed data is used to create a back-propagation neural network to classify the instances. A normal backpropagation network with a fixed number of hidden units and layers was first implemented. Later, capabilities to dynamically manipulate the number of hidden units were used [2]. A step-by-step approach of modifying the network helped in understanding the improvements made by each such modification.

Cross validation is a very useful technique in evaluating the performance of the constructed neural network model. It shows the ability of model in classifying an unknown instance. Of various cross-validation approaches such as holdout, k-Fold, leave-one-out cross validation, k-Fold was chosen since the number of instances in the dataset were few [8].

2 Method

2.1 The Dataset

Diabetic Retinopathy Debrecen (DRD) dataset contains 1151 instances that is represented with 20 attributes. The features represent detected lesion and other information of anatomical part of image-level descriptor [1]. All 20 input attributes provided were important and crucial for the successful classification of instances. The target associated with the dataset is to predict whether a given instance can be considered as a Diabetic Retinopathy or not.

As the dataset contained only 2 classes to predict and is balanced, various techniques such as clustering abundant instances to handle imbalanced data-sets and multiple classes could be ignored. The dataset could be easily identified as a balanced one with a nearly 1:1 ratio for both diabetic retinopathy and non-diabetic retinopathy targets. One of the challenges involved in pre-processing of the dataset was to normalize the values. Values of features were very distributed making it mandatory to normalize in order to avoid ill-conditioning of the model [7]. Furthermore, not having a row indicating the feature names made it complicated to access each row by name.

Data was loaded into a data-frame, which was then processed to avoid potential anomalies. In the DRD dataset, values of various features ranged from -2 to 345. Such huge distributions of values would result in additional weightage given to features with higher values. Normalization and standardization are two widely used methods to handle such cases [10]. The normalization procedure restricts values to a [0, 1] range and this was used to rescale the data.

Also, the columns of the dataset did not have labels. A row was added to name each column as F_i (where *i* indicates the column index). Adding a labelled row enabled to easily access and identify all columns. The first column of the dataset represents the quality of the image, where 0 suggests bad quality and 1 indicates good quality. As data inferred from a bad image could possibly disrupt the learning of the model, all four instances with 0 values in image quality were stripped.

Data was formatted into 20 columns, where the first 19 represents the input features and the last column represents the target values. All of the 19 input feature columns were normalized. Normalizing the target column did not make sense in a classification problem since the model requires the exact class labels.

2.2 Model

A back-propagation neural network was implemented. The network was designed with an input layer, 1 hidden layer and an output layer. The number of hidden layers could be altered to fit the dataset. More number of hidden layers generally improves the accuracy. However, this can result in a surge of test errors and an increased number of computations [11].

The number of hidden units was initialized to 500, which can be dynamically pruned using various techniques (discussed later in report). When the number of hidden units was set to 500, the input layer became a matrix of size 19x500 and the output layer matrix of size 500x1 (since it is a binary classification problem).

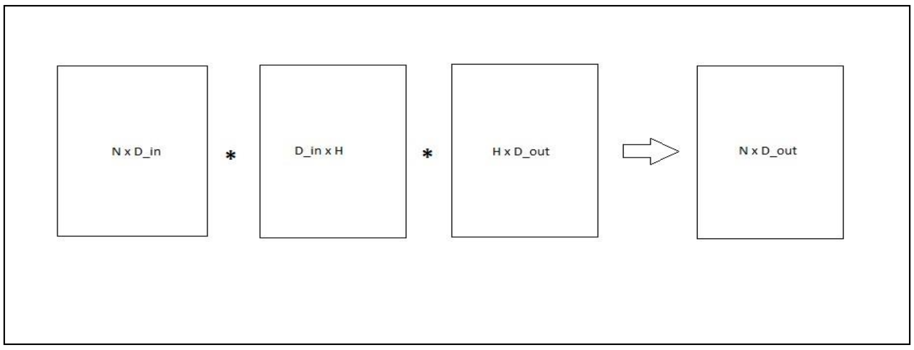

Fig. 1. Structure of model, D_in is 19, H is 500 and D_out is 1.

Two other factors that are important for a good design of a neural network model is the learning rate and the number of epochs. In a model where stochastic gradient decent is used for optimization, the learning rate parameter tells the model how far to move the weights for a batch [12]. If learning rate is low, the training becomes more reliable since there is a lesser chance to miss minima but, will be computationally more expensive as more iterations are required for the model to converge. On the other hand, if the learning rate parameter is set to a high value, the training will be less reliable and easy on computational resources.

Epochs represent the number of forward and backward passes of all instances. This means that one epoch consists of the process of feeding input to the network, predicting output, computing gradients, and updating the weights. A large number of epochs improves the training accuracy. However, similar to learning rate, such accuracy comes at the cost of computational resources. Additionally, more number of epochs could also result in the creation of an over-fitted model.

2.2.1 Implementation of mini-batch

The initial implementation was done by considering the entire training dataset at one for the number of given epochs. Mini-batch is the process of splitting the dataset into small batches to calculate the error in a batch than as a whole [3].

3 This manner of model reconditioning results in more frequent weight modifications. As a result, the model will train towards a robust convergence avoiding local minima [3].

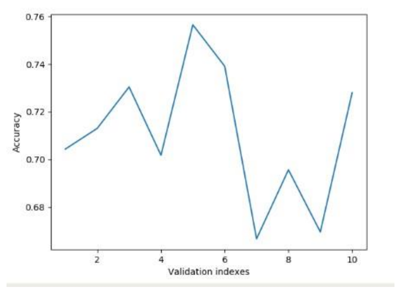	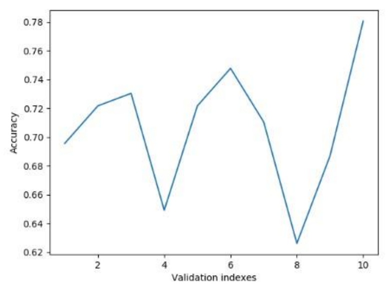
With 5 mini-batches	With entire set of instances

Fig. 2. Accuracy scores with and without mini-batch training.
Accuracies are plotted against each set of validation set used.

2.2.2 Implementation of distinctiveness

Determining the exact number of hidden neurons by looking at a data set is a close to impossible task. There is no method to clearly identify and determine an exact number of hidden units for an optimal network [13]. Various techniques such as relevance, contribution, sensitivity, distinctiveness and badness has been used to prune neural networks [15]. Of the aforementioned techniques, this paper will discuss the distinctiveness technique. The method of distinctiveness is applied by determining the similarity between each hidden unit with all other hidden units. Based on the results of distinctiveness, the network is then pruned to remove the redundant hidden neurons [15].

As suggested by *T.D. Gedeon* [15], in order to prune vectors of hidden units, an angular separation of up to 15° could be considered similar and those with over 165° can be considered complimentary. The same approach was employed in my model and comparisons were done with the cosine values of angle rather than angle itself. This form of pruning resulted in a network with fewer hidden neurons. Since only similar hidden neurons were removed, no further training on the pruned network was required [2].

There was no dip or increase in accuracy after neurons were removed. Although there were noticeable differences in values for each validation unit, the average values of all validation units persisted at around 72%.

2.2.1 Implementing early stopping

As mentioned earlier, determining the number of epochs was a challenging task. Though the accuracy of the model should improve with higher number of epochs, it turned out to produce a model that overfits on training data. Hence, halting the learning of the network before it began overfitting was essential.

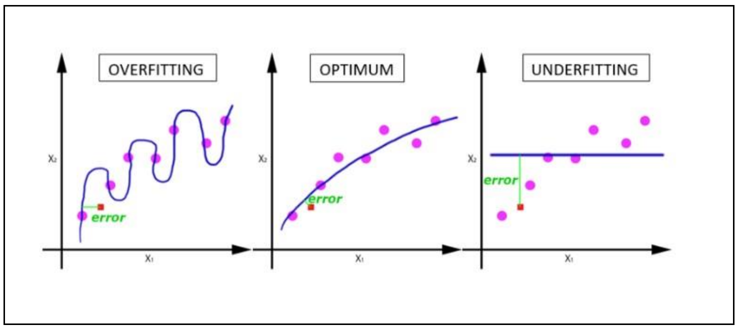

Fig. 4. Effect of various epoch values [16]

One way to prevent overfitting due to a large number of epochs is by employing a technique known as early stopping. In early stopping, the network stops training when it has stopped improving. In the created model, for every epoch after the 200th epoch, a check is made to verify whether the errors flatten out and the learning is stopped when it does.

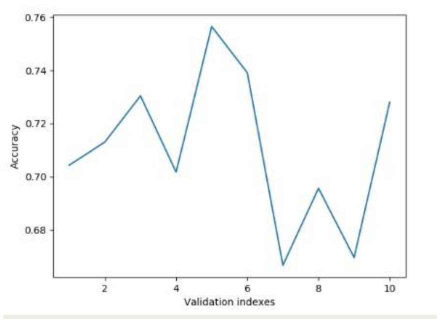	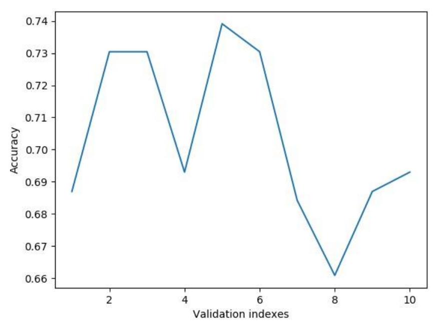
Accuracy with 1000 epochs	Accuracy with Early Stopping

Fig. 5. Accuracy scores for various epochs training.
Accuracies are plotted against each set of validation set used.

2.2.3 Implementation of Genetic Algorithm

Genetic algorithms (GA) are a heuristic search and optimization technique inspired by natural evolution [18]. One of the major advantages of genetic algorithm based optimization is that the chances of falling in local minima are greatly minimized.

Until now, the hyper-parameters used were figured out by manually trying out different combination of values and also with the help of domain knowledge. However, it is a significantly difficult task to manually try out all permutation of hyper-parameters. In situations like this, optimization using genetic algorithms are beneficial. To find the best set of hyper-parameters, we create models the random combination of hyper-parameters, select the best values and then process those further.

The entire process follows the natural selection process of reproduction. Similar to natural selection, we mutate, reproduce and create new offspring's of the best/fittest chromosomes. Each permutation of hyper-parameters is known as chromosomes and each attribute, a gene. The process is represented in the flowchart below.

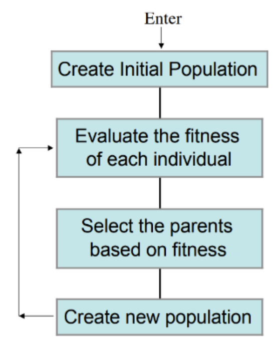
Enter
Create Initial Population
Evaluate the fitness of each individual
Select the parents based on fitness
Create new population

Fig. 6. Process: Genetic Algorithm [18]

From the flowchart, it is clear that an approach where we select the best parameters, mutate and reproduce them, and repeat the whole process again would lead to a better solution. There are several ways to determine how a chromosome can be deemed good, the scoring measure is known as fitness and the method for finding fitness, the fitness function. This paper uses the accuracy of the model as the fitness function. More attributes such as computational cost could also be considered, however, is ignored in this paper to keep the discussion simple.

The number of hidden-units, epochs, learning-rate, no of batches to be split into, and the optimization algorithm was the attributes/genes in a chromosome. Initially, the population is created by randomly picking values for each gene in a chromosome. Each chromosome is then evaluated and the fitness function is recorded. Then, the selection process where the best chromosomes are selected is done. These selected chromosomes are used to create more offsprings and mutate. As shown in the figure, this whole process of selection-mutation-breeding is repeated. Each repeating step is known as a generation.

In this paper, a chromosome is created with 5 genes, number of hidden layers, number of epochs, no of batches to be split, learning rate and the optimizer to be used. The population size is initialised with 200 to generate a wide permutation of values. After the first iteration, the size is reduced to 10 and maintained at 10 for all the following generations. For implementing crossover, uniform crossover is used as it is regarded to be a better choice than one-point and two-point crossover techniques in optimization. Also, random mutate and rank based selection is used for mutation and selection procedure. 3 chromosomes are selected in each generation for breeding and mutation.

With genetic algorithm, a new set of hyper-parameters were found. This new set of hyper-parameters were then used to generate a model and perform distinctiveness on. The accuracy of the new model increased from the previous high of 72% to 74%.

3 Results

Results of the classification of DRD dataset was obtained by using k-Fold cross-validation approach. The results are represented using confusion matrix. A confusion matrix is a technique to evaluate the results of a classification algorithm. The confusion matrix showcases how the trained model gets confused on new and unseen data when it makes predictions. Classification accuracy is also a good method to understand the performance of a classification model. However, it fails to analyse reasons for lower accuracy values. Results are explained in step-by-step order to highlight the improvements made by each approach.

True Positive 27	False Negative 22
False Positive 23	True Negative 24

With 5 epochs

True Positive 39	False Negative 3
False Positive 6	True Negative 43

With epoch number given by Genetic Algorithm

Fig 7. Confusion Matrix

Early stopping of the number of epochs reduced the excessive computation by nearly 60%, when number of epochs was initialized to 1000. On an average with early stopping, only 400 epochs were used. Another implemented feature, distinctiveness, reduces the number of hidden neurons in network by about 10% when number of hidden units are initialized to 500.

Changes to Learning rates, alters the performance of model. Best results were obtained when learning rate was set to 0.001. Of the various optimization techniques including Stochastic Gradient Decent, Adam, RProp and Adamax, Adam optimizer gave most accuracy and was hence chosen. Adam optimizer is a variant of normal stochastic gradient decent with adaptive learning rates. Learning rates are adapted based on the average first and second moment of gradient.

With the usage of genetic algorithms, we could clearly identify the best set of hyper-parameters. The values of each gene, returned by genetic algorithm remained similar to the earlier estimates after implementation early-stopping, distinctiveness, and choice of Optimizer.

Hyper-parameter/Gene	Value
Number of Hidden Neurons	467
Number of epochs	375
Learning rate	0.00158
Number of batches	3
Optimizer	Adam

Table 1. Result by Genetic Algorithm.

From the result of genetic algorithm, it is clear that we can also remove the Early-stopping procedure. Though, the hidden neurons were determined using genetic algorithms, the concept of distinctiveness was used to further prune any neurons if possible.

The implemented model fared well compared to the results of *Ramaswamy et al* [14]. With a back-propagation network, an accuracy of over 74% was consistently achieved by the model. Table 4 compares the results of the designed back-propagation network model, which uses concepts of distinctiveness, with various machine learning models of *Ramaswamy et al*. The implementation of Genetic Algorithm enabled the best selection of hyper-parameters, thereby resulting in higher accuracies.

Model	Accuracy
Designed Model	74.8
Deep-learning Model	73.59
Gradient Boosting Machines	81.13
Naïve Bayes Classifier	63.45

Table 2. Comparison of result with that in *Ramaswamy et al.*

4 Conclusion and Future work

The generated model gave decent performance on the DRD dataset. Usage of Genetic Algorithm to determine the hyper-parameters was found to be very effective. Also, techniques such as early-stopping, distinctiveness, and usage of minibatches were found to be effective. However, only testing with various other data-sets and producing similarly efficient results would let us conclude on the overall performance of implemented improvements.

Though genetic Algorithm gave the best set of hyper-parameters, it was a computationally intensive process. Better techniques to understand the better value and randomize around those than generating completely random values can be considered a future project. Implemented technique of distinctiveness involves a significant computation of $O(N^2)$. An efficient algorithm of minimizing this complexity would be beneficial. Also, rather than the naïve approach of looking for dips in loss (by a counter variable), a more sophisticated implementation can be implemented.

References

[1] "UCI Machine Learning Repository: Diabetic Retinopathy Debrecen Data Set Data Set", *UCI*, 2018. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Diabetic%2BRetinopathy%2BDebrecen%2BData%2BSet. [Accessed: 29- Apr- 2018].

[2] T. Gedeon, "Indicators of Hidden Neuron Functionality: The Weight Matrix versus Neuron Behaviour", *ANNES*, 1995.

[3] M. Li, T. Zhang, Y. Chen and A. Smola, "Efficient mini-batch training for stochastic optimization", *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14*, 2014.

[4] M. Su, "Use of neural networks as medical diagnosis expert systems", *Computers in Biology and Medicine*, vol. 24, no. 6, pp. 419-429, 1994.

[5] A. Miller, B. Blott and T. hames, "Review of neural network applications in medical imaging and signal processing", *Medical & Biological Engineering & Computing*, vol. 30, no. 5, pp. 449-464, 1992.

[6] F. Jiang, Y. Jiang, H. Zhi, Y. Dong, H. Li, S. Ma, Y. Wang, Q. Dong, H. Shen and Y. Wang, "Artificial intelligence in healthcare: past, present and future", *Stroke and Vascular Neurology*, vol. 2, no. 4, pp. 230-243, 2017.

[7] S. Saarinen, R. Bramley and G. Cybenko, "Ill-Conditioning in Neural Network Training Problems", *SIAM Journal on Scientific Computing*, vol. 14, no. 3, pp. 693-714, 1993.

[8] T. Fushiki, "Estimation of prediction error by using K-fold cross-validation", *Statistics and Computing*, vol. 21, no. 2, pp. 137-146, 2009.

[9] W. McKinney, "pandas: a Foundational Python Library for Data Analysis and Statistics", *Dlr.de*, 2011. [Online].
Available:
http://www.dlr.de/sc/Portaldata/15/Resources/dokumente/pyhpc2011/submissions/pyhpc2011_submission_9.pdf. [Accessed: 29- Apr- 2018].

[10] "Standardization vs. normalization | Data Mining Blog - www.dataminingblog.com", *Dataminingblog.com*, 2018. [Online]. Available: http://www.dataminingblog.com/standardization-vs-normalization/. [Accessed: 29- Apr- 2018].

[11] S. Huang and Y. Huang, "Bounds on number of hidden neurons of multilayer perceptrons in classification and recognition", *IEEE International Symposium on Circuits and Systems*, 1991.

[12] L. Smith, "Cyclical Learning Rates for Training Neural Networks", *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017.

[13] "How many hidden units should I use?", *Faqs.org*, 2013. [Online]. Available: http://www.faqs.org/faqs/aifaq/neural-nets/part3/section-10.html. [Accessed: 29- Apr- 2018].

[14] M. Ramaswamy, "Comparative Study of Machine Learning Algorithms for Classification of Datasets using R Programming", *Scientific Journal of Impact Factor*, 2016.

[15] T. Gedeon et al, "Network Reduction Techniques”

[16] "Epoch vs batch size", *Towards Data Science*, 2018. [Online]. Available: https://towardsdatascience.com/epochvs-iterations-vs-batch-size-4dfb9c7ce9c9. [Accessed: 29- Apr- 2018].

[17] M. Ramaswamy, "Comparative Study of Machine Learning Algorithms for Classification of Datasets using R Programming", *Scientific Journal of Impact Factor*, 2016.

[18] Benjamin Lynch, "Optimizing with genetic Algorithms in Neural Networks ", 2006. [Online]. Available: https://www.msi.umn.edu/sites/default/files/OptimizingWithGA.pdf. [Accessed: 24- May- 2018].