Feedforward Neural Nets with Hyperparameter Optimisation

Introduction

Machine learning algorithms are known to solve a problem using optimizations methods , The optimization method involves many hyper parameters and layers that decide the efficiency of its problem solving, hyperparameters and layers are responsible for fitting the model to the data . There are two kinds of hyperparameter optimization i.e. manual search and automatic methods . Manual optimization works by assigning hyperparameters and other aspects manually , there is no rule or sequence it solely depends on trial and error and intuition also the tuning of those hyperparameters is not reproducible , to counter this situation automatic hyperparameter optimization algorithms are used such as grid search and random search optimization algorithms , both of them achieves automatic tuning and can obtain optimal value but have some constraints to them, grid search efficiency decreases as more hyperparameters are tuned and range increases where as in random optimization is not reliable for training some complex models. Hyper parameter optimization achieved from Bayesian optimization algorithm outperforms any other optimization models since it is an effective method that can solve costly functions or time consuming . In this report the objective is to use the Bayesian model to fine tune the hyperparameters of a feedforward neural network using Bayesian process.

Background

Bayesian optimization is a powerful approach for the global derivative-free optimization of non-convex expensive functions. Even though there is a rich literature on Bayesian optimization, the source code of advanced methods is rarely available, making it difficult for practitioners to use them and for researchers to compare to and extend them ,The BSD-licensed python package ROBO uses Bayesian optimization to tune hyperparameters for neural networks involving large datasets (Fabolas). ROBO is a new Bayesian optimization framework that offers an easy-to-use python interface inspired by the API of SciPy [Jones et al., 2001] , another example of application of Bayesian networks comes in the form of an open source python tool available Hyperopt , it is a Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms using Bayesian optimization . Similar approach will be taken in this research project where Bayesian optimization will be implemented on a dataset of choice and cross checked against different approaches.

Objective

The primary objective of this research project is to implement a feedforward neural network algorithm that tunes hyperparameters for optimization and achieving maximum efficiency while using the dataset of choice, for this particular study Titanic dataset is being used for experimentation . This research project will work by experimenting systematically with a number of different hyperparameter configurations, e.g. the learning rate, batch size, number of hidden units, layers, etc using a systematic approach of hyperparameter optimization using Bayesian algorithms . Furthermore the neural network performance will be evaluated in different settings and compared against other approaches, such as decision trees , Naive bayes and other classifiers and supported with visualizations.

Methodology

Bayesian analysis is a statistical paradigm that starts with a prior belief that is expressed as a prior distribution . This is later on updated with the new evidence to yield a posterior belief whereas posterior belief is also a probabilistic distribution.

In deep learning models are more complex and deep with involving multiple hyperparameters and layers. Tuning it in the end makes it quite expensive to achieve , optimization of hyperparameters is achieved using Manual search algorithms , Grid search , Random search or Bayesian statistics . Random and Grid search methods do not learn from previous results and are completely uninformed by the previous evaluations. In short we end up spending a significant amount of time observing hyperparameters that may not be useful anymore. The method Bayesian algorithms implement requires incorporating prior knowledge or data about hyperparameters which includes loss and accuracy of the model and prior information to find the better optimization route for selection hyperparameters for the models , it offers an approach to model uncertainty which enables exploration and exploitation during the search and remains naturally balanced . Main steps for Bayesian Optimization involves

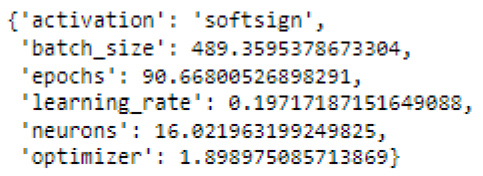
- Identifying domain of hyperparameters that require exploration for the model such as number of hidden units , dropouts, epochs, batch size , learning rate , optimizer.
- Creating an object function that takes hyperparameters as input and outputs an accuracy score. It can be loss we want to minimize or a accuracy score that we wants to maximize
- Creating a selection function for evaluation of those hyperparameters.
- Creating a history of score and hyperparameter pairs used by the algorithm during execution.

Experiment

In this experiment we are tuning 2 things of the Neural Network

1. The Hyperparameters
2. The layers

The hyperparameters tuned in this experiment are activation function, neurons , optimizers, learning rate , batch size and number of epochs to run as shown below .



```
{'activation': 'softsign',
 'batch_size': 489.3595378673304,
 'epochs': 90.66800526898291,
 'learning_rate': 0.1971718715164908,
 'neurons': 16.021963199249825,
 'optimizer': 1.898975085713869}
```

Fig. 1 Hyperparameter tuned output using Bayesian approach

The second thing to be tuned in this experiment is layers which is an additional part not required by the initial assignment but conducted to add complete value and find the maximum possible validation accuracy using the Bayesian algorithm. It is observed that different numbers of layers can affect the accuracy of the model , such as fewer layers can cause underfitting whereas many layers can cause overfitting hence affecting the performance of the model. For this experiment I am using the Titanic dataset. In this experiment ,I designed a neural network to do a binary classification task to find prediction probability of the survivors from the Titanic dataset. There are 10 columns as input from the dataset whereas two hidden layers and one output layer. In the experiment accuracy score is the accuracy metric and EarlyStopping is being used to stop the learning in case there is no accuracy gains after 20 epochs.

Tuning Hyperparameters

Neurons in each hidden layer are the same which can be made different but should be adjusted according to the nature of the solution complexity. The range of neurons in this experiment is kept from 10 to 100 which can be altered according to the nature of the dataset.

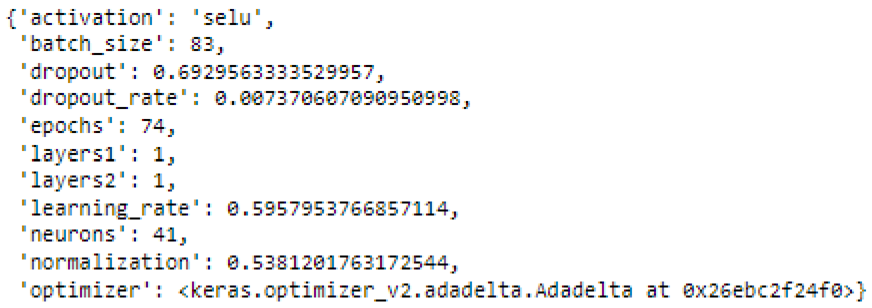
There are 9 activation functions in this experiment to be tuned **`'relu','sigmoid','softplus','softsign','tanh','selu','elu', 'exponential',LeakyReLU`**, each activation function has their own formula to compute the input values accordingly , activation functions are parameters present in each layer.

After compiling the neural network layers an optimizer is used which is responsible for adjust the learning rates and weights of neurons in the overall neural structure to find the minimum loss function , in this model there are 7 different optimizer functions used as available for selection namely **`'SGD','Adam','RMSprop','Adadelta','Adagrad','Adamax','Nadam','Ftrl'`**.

Another hyperparameter tuned is the learning rate which determines the step size for the model to reach the minimum loss function. In order to make the model more efficient in learning faster we assigned batch sizes so packs of data are given to the model at that instance , not all . Batch size is the sub-samples data meant for training for input , the smaller the batch size is the faster the learning process is , however the big batch size has learning rate which is slow in comparison but the test accuracy shows lower variance . The algorithm is designed to allow the neural network to pass through multiple epochs in order to avoid overfitting and underfitting therefore the number of epochs is tuned to gain the optimal results . The algorithm has a range of epochs from 20 to 100 respectively which can be modified based on the pool of dataset and complexity they hold. The tuning finds optimal hyperparameters using 5 fold cross validation and splits the dataset into training dataset and test dataset with a ratio of 80:20 which means the validation dataset is 20% of the total dataset according to the target variable. For Bayesian algorithm initial point are set to be 25 (`init_points=25)` and bayesian optimization steps to be 4(`n_iter=4)`

Tuning Layers

Tuning layers also plays an important role in determining the results of this model , whereas a smaller number of layers solves a simpler problem but a larger number of layers is needed to deal with a more sophisticated and complex problem , the layers are tuned using for loop interactions and are tuned between 1 to 3 layers. Regularization in this model is inserted to avert overfitting only in case if needed by the model , Batch normalization and Dropout layer are available to be used ,In Bayesian algorithm for layer tuning , initial point are set to be 25 (`init_points=25)` and bayesian optimization steps to be 4(`n_iter=4)`, below is an example from code to show the output of the layer tuning algorithm.



```
{'activation': 'selu',
 'batch_size': 83,
 'dropout': 0.6929563333529957,
 'dropout_rate': 0.007370607090950998,
 'epochs': 74,
 'layers1': 1,
 'layers2': 1,
 'learning_rate': 0.5957953766857114,
 'neurons': 41,
 'normalization': 0.5381201763172544,
 'optimizer': <keras.optimizer_v2.adadelta.Adadelta at 0x26ebc2f24f0>}
```

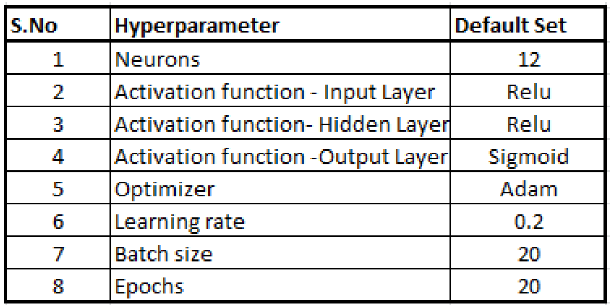
Fig. 2 Layers tuned for the selected model displaying output

Decision Tree - Naive Bayes

Finally the accuracy extracted from neural networks after filtering the hyperparameters through Bayesian algorithms is obtained , it is compared with the accuracies from decision tree , naive bayes and random forest classifiers all of them are imported from the scikit-learn library which is a tool for predictive data analysis . In all the other classifiers the test train split remains the same however in naive bayes a variant Gaussian normal distribution is used to find the validation accuracy by computing the accuracy score using validation score and predictions.

Manually Tuned Neural Network

To check the performance of Bayesian neural network optimizer it was necessary to include manually tuned neural networks working on the splitting and finding accuracies of the same dataset . The table shown below is the hyper parameter used for building the default sequential model of a neural network to perform the test and training of the dataset for further analysis and evaluation .

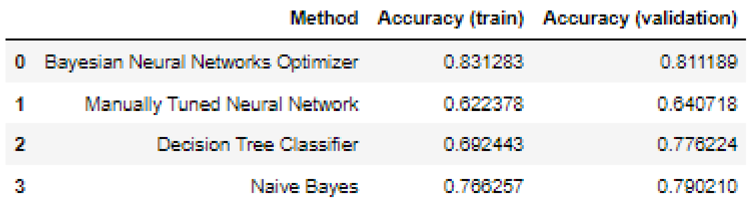


S.No	Hyperparameter	Default Set
1	Neurons	12
2	Activation function - Input Layer	Relu
3	Activation function- Hidden Layer	Relu
4	Activation function -Output Layer	Sigmoid
5	Optimizer	Adam
6	Learning rate	0.2
7	Batch size	20
8	Epochs	20

Fig. 3 Table showing the hyperparameters used to make the model for manually tuned neural network.

Results

The results achieved from the experiment gave four sets of training and validation accuracies depicting the performance of the Bayesian optimizer , Manual Tuned Neural Network , Naive Bayes and Decision Tree classifier model as shown below.



	Method	Accuracy (train)	Accuracy (validation)
0	Bayesian Neural Networks Optimizer	0.831283	0.811189
1	Manually Tuned Neural Network	0.622378	0.640718
2	Decision Tree Classifier	0.692443	0.776224
3	Naive Bayes	0.766257	0.790210

Fig. 4 Table showing the final accuracies of Bayesian optimizer , Manual Tuned Neural Network , Naive Bayes and Decision Tree classifier model

Here Bayesian Neural Networks Optimization shows a better performance in terms of training and validation compared to other used methods. Looking closely Bayesian Neural Networks Optimizer did a more deeper training and achieved an accuracy of 83% whereas the next best is Naive Bayes achieving training accuracy of 76% for the same dataset . All the used methods have a slight degree of overfitting for the dataset used even a normal neural network tuned manually however looking closely Bayesian neural network optimizer method has a balanced fitting of data compared to other methods.

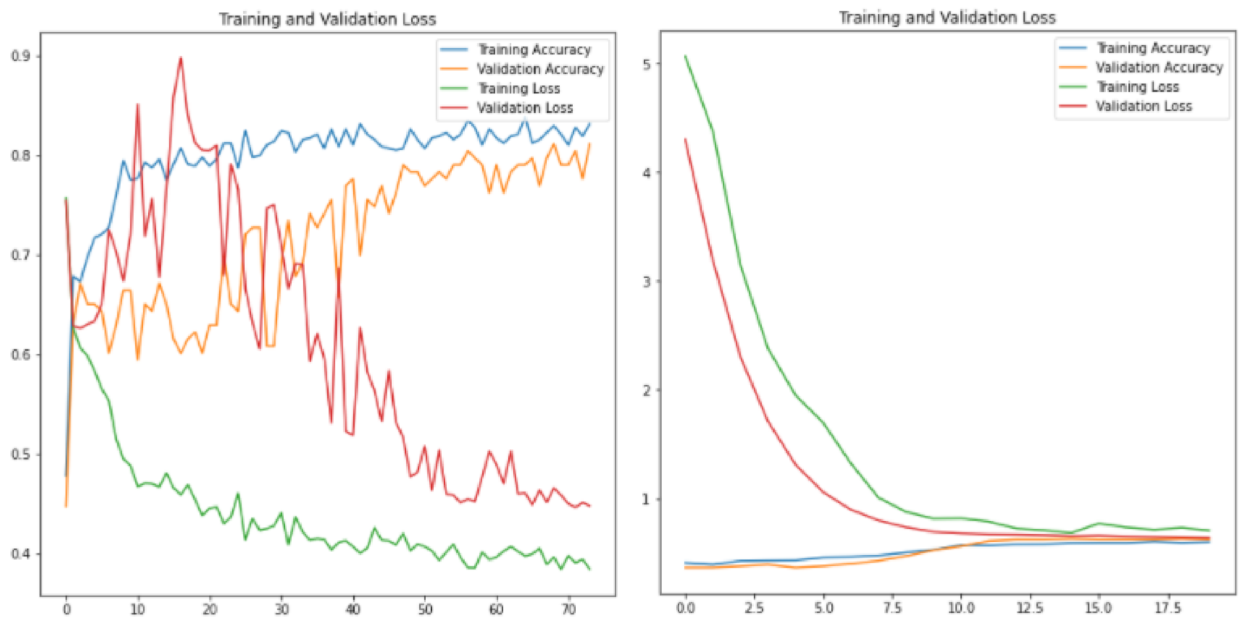


Fig. 5 Showing the final accuracies comparison of Bayesian optimizer and Manual Tuned Neural Network

Compared with the baseline set for this experiment 75% Bayesian optimization achieved beyond the set baseline . after analyzing the Training and Validation loss graph it was found that after a certain number of epochs the losses would significantly start increasing , in our case after 74 epochs and also resulting in a significant decrease of accuracy for the test dataset , this is worth further investigating as could this be extended or achieved a better accuracy if the dataset pool is increased or merged with another dataset and then evaluated again.
Apart that looking at the Training and Validation loss graph significant fluctuations are observed where there is a constant rift to achieve greater accuracy in Bayesian model , although a 83% test and 81 % validation accuracy is achieved but looking at the graph a normalization or steadiness of the loss and accuracies are not traced compared to a manually tuned neural network .

Conclusion

In this report a feedforward neural network with hyperparameter optimization was implemented and included Bayesian approach for the hyperparameter tuning of the database “Titanic” , this method experimented with a number of hyperparameter configurations while the performance was evaluated against different settings and compared against other popular approaches like decision trees and naive bayes . The preferred feedforward neural network which is Bayesian optimizer was fully implemented the effectiveness of the optimizer was challenged by the above mentioned classifiers and proved to be the best and resulted in finding better hyperparameters significantly faster than the approaches used and surpasses greatly human expertise in selecting hyperparameters for a neural network as demonstrated by comparing it with a manually tuned neural network.

References

1. RoBO: A Flexible and Robust Bayesian Optimization Framework in Python
2. Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms
3. Bayesian Optimization
4. Tuning the Hyperparameters and Layers of Neural Network Deep Learning
5. Bayesian Optimization
6. https://proceedings.neurips.cc/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf