EEE 511 - Assignment 4 Time Series Forecasting Using Multilayer Perceptron and Ensemble Learning

Xiao Huang, Suhas Lohit, Rakshit Raghavan
1206424709, 1206451918, 1207751060

***Abstract*—In this assignment, we use the multi-layer perceptron as well as an ensemble learning technique in order to perform time series forecasting. That is, given the present and few past values of a quantity e.g., stock price of a company, we predict its value in a future time instant. For training, we use the time series provided to us. For the multi-layer perceptron, we use the Levinburg-Marquardt algorithm to perform the backpropagation. For the ensemble learning model, we use bootstrap aggregating (bagging) by training multiple multilayer perceptrons with different initializations, different number of hidden neurons etc.**

***Index Terms*—Time series forecasting, multi-layer perceptron, ensemble learning, bagging.**

I. Introduction

GIVEN its ubiquitous nature, time series forecasting is a problem that has received a huge amount of attention for many decades in many fields. By a *time series*, we mean a set of measurements of a quantity or quantities made in successive instants of time over an interval. Time series forecasting is then the problem of predicting those quantities at future time instants based on the previously observed values. Making a reasonable guess can have far reaching consequences depending on the application. Some of the most important applications include weather forecasting, earthquake prediction and stock market prediction. Clearly, the hardness of the problem depends on the application and how much knowledge we have of the variables that influence the quantity that we are trying to predict. For example, it may be possible to provide a more accurate prediction of the temperature at a particular location over the next hour than predicting the stock price of a company.

Time series forecasting can be viewed in multiple ways. In particular we can consider it as a function approximation problem. Thus, we are trying, based on set of past observations (the training data), to learn a function that maps the current state of the quantity to a future state. This mapping can then be used on the testing data or deployed in the real world.

In this assignment, we use two methods to perform time series forecasting – (1) a single multilayer perceptron and (2) ensemble learning which combines predictions of multiple learning models in order to improve the performance. In particular, we use boosting to combine the predictions of many multilayer perceptrons trained with different initializations of weights, different number of hidden neurons etc. In both cases, we use the same time series provided to us for training the model. Cross-validation is also employed in both cases.

The report is organized as follows. In Section II, we describe the multi-layer perceptron approach for time series forecasting. In Section III, we discuss how we can improve the prediction using ensemble learning methods. In Section IV, the training data is visualized and the training protocol is explained in detail. Section V describes how to use the program submitted to test the performance on a test set.

II. Multilayer Perceptron

Multilayer perceptron [1] is a feedforward neural network with at least one hidden layer. The hidden layer is the reason that the network has the ability to learn complex non-linear functions as opposed to single layer perceptrons that can learn only linear functions. Theoretically, the universal approximation theorem tells us that as long as we have sufficient number hidden neurons and the non-linearity is a monotone increasing function, a single hidden layer neural network has the ability to learn an approximation to any non-linear continuous function that maps the input vector to the desired output with arbitrarily small approximation error and the error is mainly dependent on the number of hidden neurons. In practice however, we cannot have a large number of neurons since it might give rise to overfitting and the training time might be too long. Even so, the multilayer perceptron architecture is naturally suited for time series forecasting because, here too, we are learning a mapping from present and past values of a time series to a future value.

In this assignment, we use a single hidden layer network although in recent years, deep neural networks containing many hidden layers have been shown to be superior to single hidden layer networks with the same number of parameters. Such deep networks require lot of training data and are otherwise prone to overfitting. This is the reason we limit ourselves to a single hidden layer architecture.

Now, we will discuss how the multilayer perceptron can be employed for time series prediction. This is done using a windowing method. First, the number of past samples used to predict the next sample is set. For example, let us use 2 previous samples to predict the next sample. Then, during training, the input is a $2D$ vector $[x(t-1), x(t-2)]$ and the output is $x(t)$. This is repeated for all such windows in the training time series data. The error is calculated as the mean square error, over all time instants, between the network output $y(t)$ and the desired output $x(t)$. That is, training error E_{train} is given by

$$E_{train} = \frac{1}{T}\sum_{t=3}^{T}(y(t)-x(t))^2 \quad (1)$$

where $y(t)$ is clearly a non-linear function of the input $[x(t-1), x(t-2)] \in \mathbb{R}^2$ and T is the total length of the training data. Backpropagation is used to minimize this error by tuning the weights in the network. There are many variants of backpropagation. In this part of the assignment, we use the Levenberg-Marquardt algorithm. This is a very fast algorithm if the dataset is small and tends to provide better results than simple gradient descent. This is also an attractive option since the learning rate is automatically set.

We also use Bayesian regularization, another variant of backpropagation, later for ensemble learning (See Section III). In regularization, we modify the error function in Equation 1 to include another term, which is usually the mean square of the weights and biases of the network. Thus the regularized training error becomes

$$E_{train-reg} = \frac{1}{T}\sum_{t=3}^{T}(y(t)-x(t))^2 + \lambda\frac{1}{M}\sum_{i=1}^{M}w_i^2 \quad (2)$$

where M is the total number of weights w_i. Here, the parameter λ must be set manually. Bayesian regularization improves this by estimating it statistically using a maximum a posteriori (MAP) estimate [5].

For testing, there are two options – open loop and closed loop prediction. In open loop prediction, the predictions $y(t)$ are only one step ahead, where the predictions are based on previous values of $x(t)$. In closed loop prediction, $y(t)$ are obtained by previous predictions of the network itself rather than the true values $x(t)$. We use open loop prediction for testing since it is more accurate, hence the ground truth testing data must be provided in order to test the network.

III. Ensemble Learning

In statistics and machine learning, ensemble methods [2] use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms. In other words, in the ensemble learning approach, we train multiple classifiers or predictors and then combine them to get better performance in prediction, compared to a single model.

The main motivation for ensemble learning is the hardness of model selection. In real world pattern recognition problems, we often have access to a handful of training data which may or may not be representative of the true distribution of the data. In such a case, an algorithm that simply minimizes the training error may not be beneficial. The question then is of selecting the right model from a list of models that may result in similar training error. With no further knowledge about the data, it may seem that selecting a model at random might be a good choice. But, this increases the risk of selecting a bad model. Instead, combining the predictions of different models has been shown to be much more effective.

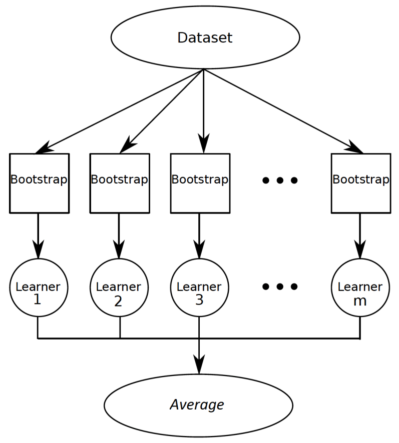

Fig. 1. Illustration of the bagging algorithm for time series forecasting. Here, each learner is a multilayer perceptron.

There are various algorithms available for ensemble learning. In this assignment, we focus on one type of ensemble learning called bootstrap aggregating, also known as bagging [3]. Here, multiple models are trained on randomly sampled (with replacement) subsets of the training dataset. The final output is the combination of outputs of the individual models. This is either obtained through voting in the case of classifiers or by averaging in the case of prediction. Other algorithms include boosting where the classifiers are trained in succession by resampling, and the resampling is done intelligently in order to provide the most informative data for the next classifier, Mixture of Experts where the weights of the combination are learned using the EM algorithm etc.

Following are steps to be followed for bagging:

1) Given a dataset D with T datapoints, generate M new subsets $D_i, i = 1, , M$ by sampling randomly with replacement. Data points in D_i can either be appear in D_j or not.
2) Train M classifiers or predictors $P_i, i = 1, , M$. In our case, each predictor is a multilayer perceptron.
3) Combine the outputs of the M predictors. Many options exist such as majority voting (for classification tasks), weighted majority voting, mean rule, median rule etc [2]. We use the mean rule where the final ensemble output is simply the mean of the M individual outputs.

For testing, the predictions from all the trained models are averaged to give the final prediction. In our case, the weak learner is the multilayer perceptron described in Section II with trained with different parameters such as initialization, backpropagation algorithm, number of hidden neurons etc.

IV. Training and Testing Protocol

The given training data is a 276 point time series where samples are taken one year apart. The training data is shown in Figure2.

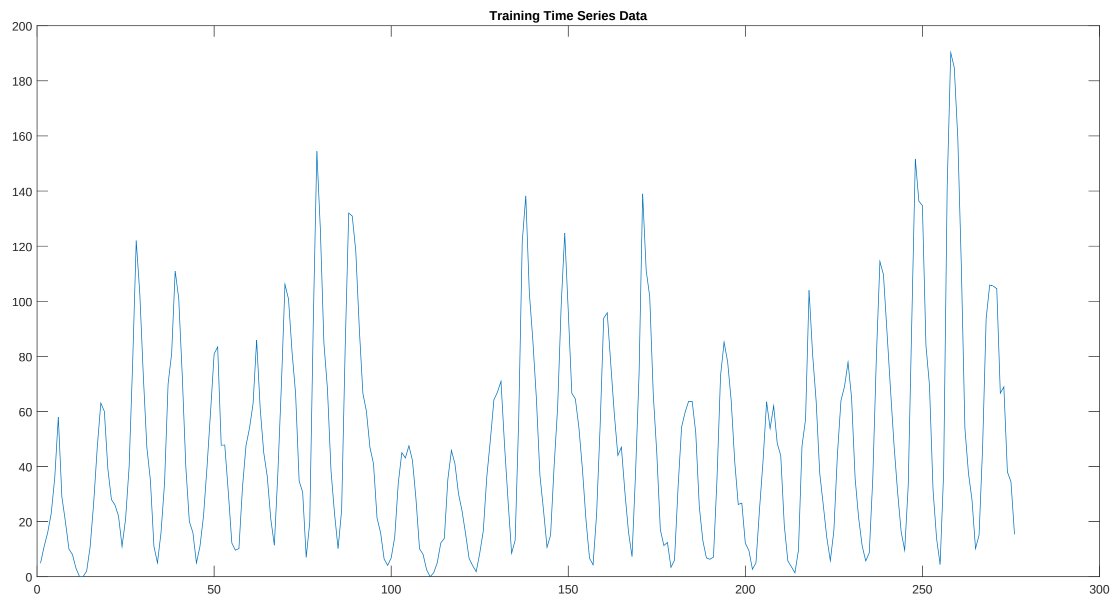

Fig. 2. Time series data used for training.

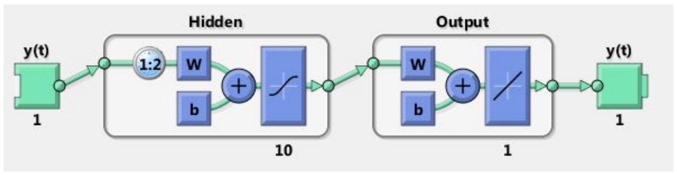

Fig. 3. Network architecture used for part 1 of the assignment (generated by Matlab).

A. Multilayer perceptron

In the first part, we train single multilayer perceptron for time series forecasting using the given training dataset. The network that we use uses 3 previous samples to perform prediction (this is indicated in the variable `feedbackDelays`) in the code. The hidden layer has 10 neurons and the output layer has a single neuron that outputs the prediction. There is no nonlinearity used in the final layer since we want $y(t) \in \mathbb{R}$. The diagram of this network is shown in Figure 3. The weights are initialized randomly and Levenberg-Marquardt algorithm is used for backpropagation. 80% of the data, sampled randomly is used for training and the remaining 20% is used as cross validation. Backpropagation is stopped when the cross-validation error starts increasing, since this is a sign of overfitting.

Note: Since, we do not have the testing data, we ”tested” the network using the same training data. The plot in figure 4 shows the results. **This is done only to verify the correctness of the program and is not indicative of the actual performance of the network on the actual test data**.

Note that the output predictions are quite noisy and the error in prediction is also quite high. In order to improve the performance, we use ensemble learning as explained in the next section.

B. Ensemble Learning

As mentioned in Section III, we use bagging as the ensemble learning method. We train multiple neural network models with different settings of parameters and finally average the results of individual networks to output the prediction of the ensemble model.

We train 36 different models. Other than for initialization, at most three parameters are changed between models:

1) Backpropagation algorithm: Levenburg-Marquardt or Bayesian Regularization.
2) Number of hidden neurons : $10, 12$ or 15.
3) Input dimension, i.e., the number of previous samples used to make the prediction: $2, 3$ or 4.

For each of the 12 combinations of the above parameters, 3 networks are trained with different random initializations, thus bringing the total number of models to 36.

As before, we use open loop prediction for testing, and hence the testing data needs to be input to the code. Each of the trained networks outputs its prediction and ensemble model output is simply the mean of the individual predictions.

Note: As before, since, we do not have the testing data, we ”tested” the network using the same training data. The plot in figure 5 shows the results. **This is done only to verify the correctness of the program and is not indicative of the actual performance of the network on the actual test data**.

V. Using the code submitted for testing

The code submitted is written in Matlab 2014b and uses functions from the Neural Network toolbox (available in MyApps). Before calling the function, make sure that

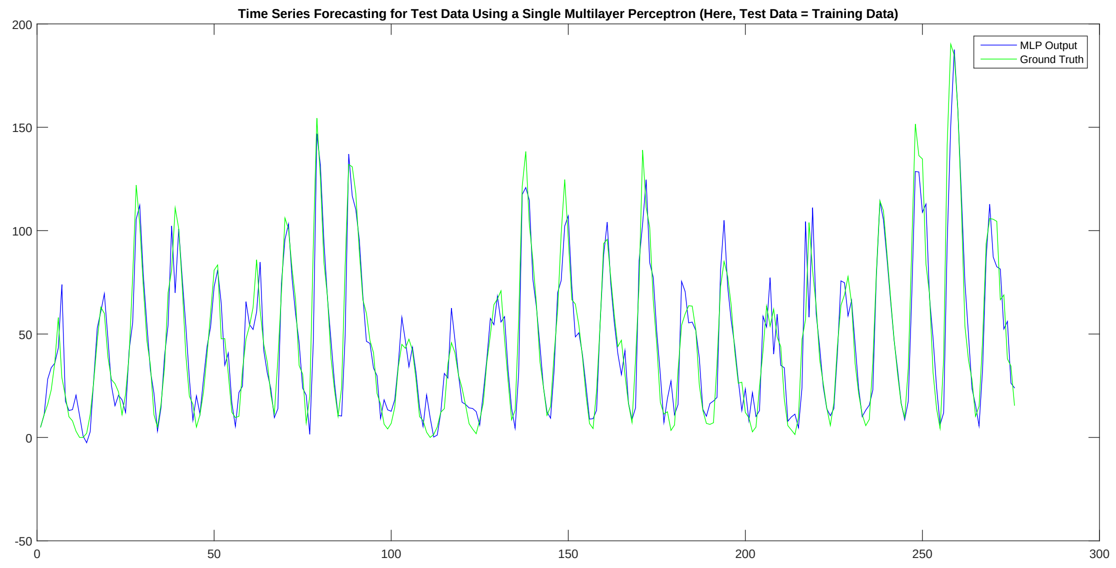

Fig. 4. Verifying the correctness of the multilayer perceptron training and testing code. The plot shows the training time series used as the testing data and the network predictions when asked to predict this time series using open loop method.

the ground truth test data, in the form of an array (not a cell array) is loaded into the Matlab workspace and the `train_data.mat` provided is also present in the same directory. Call the function using the following line: `[prediction_mlp, prediction_ensemble, MSE_mlp, MSE_mlp_percentage, MSE_ensemble, MSE_ensemble_percentage] = time_series_forecast(<test data variable name>);`. This will return the predictions for both parts of the assignment and will also plot them against the ground truth test data. Read the comments in the code for more details.

VI. Conclusion

Time series forecasting is an important machine learning prediction problem and to solve this, we have used two approaches – the first being the multilayer perceptron and the second being an ensemble learning technique using bootstrap aggregating (bagging). Learning in the multilayer perceptron is done using backpropagation wherein training error is minimized with respect to the weights in the network to achieve the desired result. The Levenberg - Marquardt backpropagation algorithm was applied due to its simplicity, ease of application and robustness. The effectiveness of ensemble learning was tested on the same time series dataset and it is found that this method provides better results since this is a combination of predictors and is more accurate in comparison with an individual learning model.

References

[1] Simon Haykin, *Neural Network and Learning Machines*, Vol. 3. Upper Saddle River, Pearson Education, 2009.

[2] Robi Polikar, *Ensemble Learning*, Scholarpedia Link

[3] L. Breiman, *Bagging predictors*, Machine Learning, vol. 24, no. 2, pp. 123-140, 1996

[4] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, *Adaptive mixtures of local experts*, Neural Computation, vol. 3, pp. 79-87, 1991.

[5] *Backpropagation*, UFLDL Link.

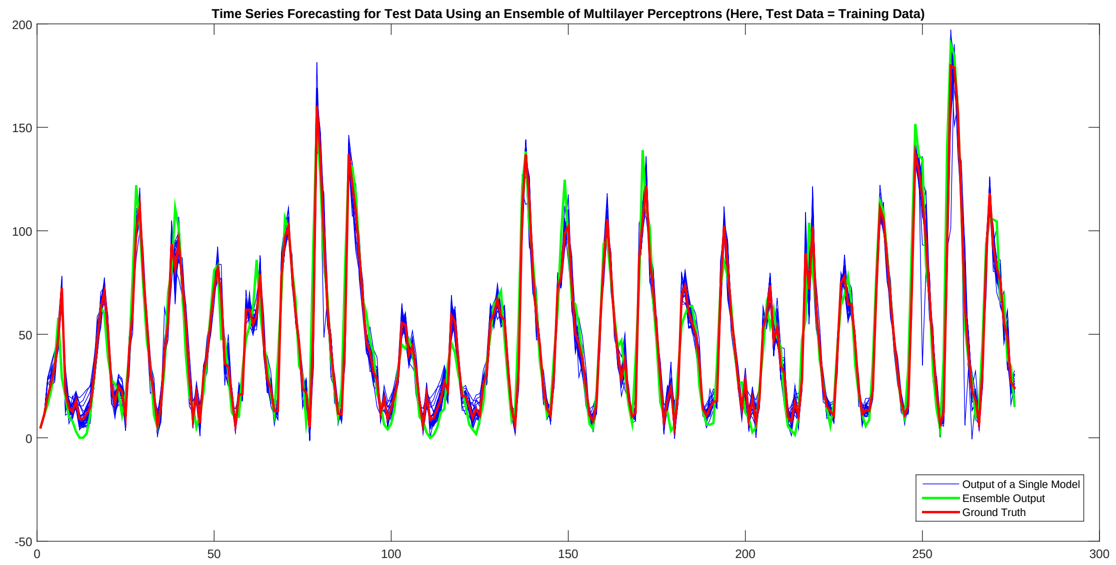

Fig. 5. Verifying the correctness of the ensemble learning training and testing code. The plot shows the training time series used as the testing data and the ensemble model predictions when asked to predict this time series using open loop method.