A Combined Neural Network/Gaussian Process Regression Time Series Forecasting System for the NN3 Competition

Nesreen K. Ahmed
Faculty of Computers and Information
Cairo University, Giza, Egypt
n.kamel@gmail.com

Amir F. Atiya
Dept Computer Engineering
Cairo University, Giza, Egypt
amir@alumni.caltech.edu

Neamat El Gayar
Faculty of Computers and Information
Cairo University, Giza, Egypt
hmg@link.net

Hisham El-Shishiny
IBM Center for Advanced Studies in Cairo
IBM Cairo Technology Development Center
Giza, Egypt
shishiny@eg.ibm.com

1 Introduction

In a recent study [1] we have conducted a large scale empirical comparison of seven different machine learning models for time series forecasting using the M3 benchmark data. The outcome of this study is that the standard multilayer perceptron neural network (MLP) and Gaussian process regression (GPR) have turned out to be respectively the first and the second best methods. Taking cue from this study, we propose here a combined model that applies MLP and GPR using a number of different input preprocessing methods. The model combination includes some aspects of forecast combination using simple averaging and model selection based on the training set and validation set performance. What makes this competition challenging is the large forecast horizon (the next eighteen points have to be forecasted). The study in [1] was simply for the one-step ahead case. We have devised input preprocessing steps that seem to be suitable for different forecast horizon ranges. In addition to the machine learning part, we have carried out a thorough preprocessing of the time series, including deseasonalization (if needed), log transformation, and scaling. Next section gives a description of the MLP model and the GPR model, while Sections 3 and 4 provide the details of the forecasting system. Section 5 details the overall set-up, and Section 6 gives the validation results of the developed models.

2 Models

2.1 Multilayer Perceptron (MLP)

The multilayer perceptron (often simply called neural network) is perhaps the most popular network architecture in use today both for classification and regression. The network has a simple interpretation as a form of input-output model, with the weights and biases being the free parameters of the model. Such network can model functions of arbitrary complexity, with the number of hidden nodes determining the network complexity. As is typically the practice, we have considered the traditional one-hidden layer network. Instead of using the well-known backpropagation algorithm for training, we have considered a second order optimization method called Levenberg Marquardt, as this is generally known to be more efficient (we use the Matlab function `trainlm`). The number of hidden nodes is a critical parameter for MLP, as it determines the network complexity. We have used five-fold validation to estimate the most suitable number for every time series. We consider the candidate values $NH = [0, 1, 3, 5, 7]$. Note that we have the possibility of “having zero hidden nodes” ($NH = 0$), meaning simply a linear model. Balkin and Ord [2] have shown that the possibility of switching to a linear model for some time series improved performance. Concerning the other less key parameters and model details, we selected them as follows. We have used the sigmoidal activation functions for the hidden layer, and a linear output layer. Training is performed for 500 epochs, using a momentum term 0.2, and an adaptive learning rate with initial value 0.01, an increase step of 1.05 and a decrease step of 0.7.

2.2 Gaussian Processes (GP)

Gaussian process regression [5] is based on modeling the observed responses of the different training data points (function values) as a multivariate Gaussian random variable. For these function values an a priori distribution is assumed that guarantees smoothness properties of the function. Specifically, the correlation between two function values is high if the corresponding input vectors are close (in Euclidean distance sense) and decays as they go farther from each other. The posterior distribution of a to-be-predicted function value can then be obtained using the assumed prior distribution using simple probability manipulations.

Let $V(X, X)$ denote the covariance matrix between the function values, where X is the matrix of input vectors of the training set (let the $(i, j)^{th}$ element of $V(X, X)$ be $V(x_i, x_j)$, where x_i denotes the i^{th} training input vector). A typical covariance matrix is the following:

$$V(x_i, x_j) = \sigma_f^2 e^{-\frac{\|x_i - x_j\|^2}{2\alpha^2}} \tag{1}$$

In addition, some independent zero-mean noise having standard deviation σ_n is assumed to be added to the function values to produce the observed responses (target values).

Then, for a given input vector x_*, the prediction $\hat{y}_*$ is derived as

$$\hat{y}_* = E\big(y_* | X, y, x_*\big) = V(x_*, X)\big[V(X, X) + \sigma_n^2 I\big]^{-1} y \tag{2}$$

where y is the vector of target outputs (response values) for the training set. For Gaussian processes there are three key parameters: σ_f, σ_n, and α. It will be prohibitive to use a three-dimensional ten-fold validation approach on these parameters. We opted for the model selection algorithm proposed by Rasmussen and Williams [5]. It is an algorithm that maximizes the marginal likelihood function. The authors make a point that such criterion function does not favor complex models, and overfitting will therefore be unlikely (unless there is a very large number of hyperparameters).

3 Input Preprocessing

Let $x(t)$ be the time series value at step t, and let the forecast horizon be h. That is we would like to forecast $x(t+h)$ (not the whole range: $x(t+1),\ldots,x(t+h)$). We considered the following inputs that are extracted from the time series:

1. Lagged time series values (LAGGED-VAL): the inputs to the prediction model are the lagged time series values $x(t-(N-1)h),\ldots,x(t-h),x(t)$. The reason we go back in steps of h is that we would like to keep up with time resolution dictated by the forecast horizon.

2. Taking moving averages (MOV-AVG): We compute moving averages over rectangular smoothing windows placed at different time instants in the period previous to t:

$$u_i(t)=\frac{1}{h}\sum_{j=t-ih}^{t-(i-1)h-1}x(j), \qquad i=1,2,\ldots \quad (3)$$

   in addition to given $x(t)$ as input as well. The reason we add $x(t)$ as input is that moving averages inspite of their nice smoothing characteristics introduce a harmful lag effect. The addition of $x(t)$ will make the given information as recent as could be achieved. We consider moving averages up to a certain specified overall lag window J (for example for a lag window of 24 months, we keep using delayed moving averages until we reach the end of the lag window, so the lag window will dictate the number of moving averages, i.e. the number of inputs we use).

3. Exponential moving averages (EXP-MOV-AVG): Define an exponential moving average for a fixed window of data starting from $t-J+1$ to $t-i$:

$$z_i(t)=\sum_{j=t-J+1}^{t-i}a^{t-i-j}x(j)/D \quad (4)$$

   where a is the decay factor (it should be in the range from 0 to 1) and D is the factor that makes all weights sum to 1. For this group of inputs, we considered: the following: $x(t), z_1(t), z_h(t), z_{2h}(t), \ldots$. Also here J denotes the overall lag window. Beyond that we do not take any more time series values. Again, we added the most recent time series value $x(t)$ to avoid the delay issue.

For only forecast horizons from 1 to 11 we add an additional input to each of the above input combinations. That input is $x(t+h-12)$, which represents the time series value 12 months before the time of the data point to be forecasted. This helps to account for whatever seasonality might exist. Even though we perform a deseasonalization step, this input is helpful as some time series have a weak seasonality component that that they did not pass the seasonality test. Also, the process of deseasonalization is far from perfectly accomplished and some seasonal trace is usually present.

4 Other Preprocessing

Upon quick inspection of the time series, we have found that many of them possess seasonality. We have therefore performed a deseasonalization step for those time series that pass the seasonality test. In addition, to get the time series to be in a suitable level and range we have applied a log transformation and scaled the data. In summary, we perform the following transformation, in the following order:

1. Log transformation

2. Deseasonalization

3. Scaling

For the log transformation, we simply take the *log* of the time series. Concerning deseasonalization, a seasonality test is performed first to determine whether the time series contains a seasonal component or not. The test is by taking the autocorrelation with lag 12 months, to test the hypothesis "no seasonality" with using Bartlett's formula for the confidence interval (Box and Jenkins [3]). If the test indicates the presence of seasonality, then we use the classical additive decomposition approach [4]. In this approach a centered moving average is performed, then a month-by-month average is computed on the smoothed series. This average will then be the seasonal average. We subtract that from the original series to create the deseasonalized series. The scaling step is essential to get the time series in a suitable range, especially for MLP where scaling is necessary. We have used linear scaling computed using the training set, to scale the time series to be between -1, and 1.

After these transformations are performed we extract the input variables (LAGGED-VAL, EXP-MOV-AVG, or MOV-AVG) from the transformed time series. Then the forecasting model is applied. Once we perform the forecasting, we unwind all these transformations of course in reverse order.

5 The Overall Set-Up

First we check the length of the time series. If it is ≥ 60 data points then we use the following design.

We have developed a separate prediction model for each of the 18 forecast horizons. The prediction model uses for most of the horizons two different input preprocessing methods (one separate prediction model for each of the two input preprocessing methods). The forecasts of the two prediction models are combined using simple averaging. This is repeated for MLP as well as GP, leading to the "MLP forecast" and the "GP forecast". Furthermore, the MLP forecast and the GP forecast are combined using simple averaging, leading to the "Combined forecast". Based on a combination of a validation set performance and the training set performance (both equally weighted) for the considered time series, we choose the final model from among the three candidates: the MLP forecast, the GP forecast, and the combined forecast. (We have withheld 18 points for each time series for validation, but once the validation error is computed the model is retrained using all data of the considered time series.) The winning candidate is the one that we used to obtain the final forecast for the 18 out-of-sample points. The input preprocessing methods used are as follows:

1. For one-step ahead, we use only LAGGED-VAL preprocessing. Using K-fold validation we choose the best number of lags (up to 6) to take.

2. For forecast horizons from 2 to 7 we use LAGGED-VAL preprocessing and EXP-MOV-AVG preprocessing, with the overall lag window (J as defined in Section 3) equal to 12.

3. For forecast horizons from 8 to 18 we use MOV-AVG preprocessing and EXP-MOV-AVG preprocessing, with the overall lag window J equal to 24.

If the time series is too short (< 60 data points), then there is no point to apply the machine learning methods as even the most concise model will lead to some kind of overfitting. In such a case we use a simple linear regression. This applies to the time series numbers 1 to 50 (let us call them Group 1, while the longer time series, numbered 51 to 111, is Group 2). We also did not implement the described preprocessing steps such as deseasonalization, because the seasonality issue is taken into account by the choice of the inputs (see below). Also, by inspection we found that the seasonality for this group is weak if at all there is. The inputs to the linear regression are the following:

1. The most recent time series value $x(t)$.

2. The moving average of the time series over a window from $t-L+1$ to t, where L depends on the forecast horizon and varies from 9 to 15.

3. For only forecast horizons from 1 to 11 we use a third input, and that is $x(t+h-12)$ which represents the time series value 12 months before the time of the data point to be forecasted. This helps to account for whatever seasonality might exist.

6 Validation results

We withheld from the training set 5 points for Group 1 and 18 points for Group 2 for the purpose of validating the performance of the methods. We used as error measure the symmetric mean absolute percentage error, defined as

$$SMAPE = \frac{1}{M}\sum_{m=1}^{M}\frac{|\hat{y}_m - y_m|}{(|\hat{y}_m| + |y_m|)/2} \qquad (5)$$

where y_m is the target output and $\hat{y}_m$ is the prediction. Since it is a relative error measure it is possible to combine the errors for the different time series into one number. The validation SMAPE for Group 1 turned out to be 16.11% while the validation SMAPE for Group 2 turned out to be 15.66%.

Acknowledgement

We would like to acknowledge the help of Athanasius Youhanna of Cairo University, who has developed the seasonality test and the trend test for this work. We also would like to acknowledge the useful discussions with Professor Ali Hadi of the American University of Cairo and Cornell University. This work is part of the *Data Mining for Improving Tourism Revenue in Egypt* research project within the Egyptian Data Mining and Computer Modeling Center of Excellence.

References

[1] N. K. Ahmed, A. Atiya, N. El Gayar, and H. El-Shishiny, *An empirical comparison of machine learning models for time series forecasting*, Accepted in the Special Issue on *The Link Between Statistical Learning Theory and Econometrics: Applications in Economics, Finance, and Marketing*, Econometric Reviews, to appear on 2009.

[2] S.D. Balkin and J.K. Ord, *Automatic neural network modeling for univariate timeseries*, International Journal of Forecasting, 16(4), 509-15, 2000.

[3] G. Box and G. Jenkins, *Time Series Analysis, Forecasting and Control*, Holden-Day Inc., 1976.

[4] S. Makridakis, S. C. Wheelwright, and R. J. Hyndman, *Forecasting: Methods & Applications*, 3rd Eddition, Ch. 3, Wiley, 1998.

[5] C. E. Rasmussen, and C. K. L. Williams, *Gaussian Processes for Machine Learning*, MIT Press, 2006.