PSF : Introduction to R Package for Pattern Sequence Based Forecasting Algorithm

Article · June 2016

2 authors:

Neeraj Dhanraj Bokde
Visvesvaraya National Institute of Technology
13 PUBLICATIONS **0** CITATIONS
SEE PROFILE

K.D. Kulat
Visvesvaraya National Institute of Technology
62 PUBLICATIONS **83** CITATIONS
SEE PROFILE

PSF : Introduction to R Package for Pattern Sequence Based Forecasting Algorithm

by Neeraj Bokde and Kishore Kulat

Abstract This paper discusses about **PSF**, an R package for Pattern Sequence based Forecasting (PSF) algorithm used for univariate time series future prediction. The PSF algorithm consists of two major parts: clustering and prediction techniques. Clustering part includes selection of cluster size and then labeling of time series data with reference to various clusters. Whereas, the prediction part include functions like optimum window size selection for specific patterns and prediction of future values with reference to past pattern sequences. The **PSF** package consists of various functions to implement PSF algorithm. It also contains a function, which automates all other functions to obtain optimum prediction results. The aim of this package is to promote PSF algorithm and to ease its implementation with minimum efforts. This paper describe all the functions in **PSF** package with their syntax and simple examples. Finally, the usefulness of this package is discussed by comparing it with *auto.arima*, a well known time series forecasting function available on CRAN repository.

Introduction

PSF stands for Pattern sequence based forecasting algorithm. PSF is a forecasting technique based on pattern sequences present in the time series data. For the first time, it was proposed in Martínez-Álvarez et al. (2008). An improved version was discussed in Martinez Alvarez et al. (2011).

The PSF algorithm consists of many processes, which can be divided broadly in two steps. The first step is clustering of data and second step is forecasting based on clustered data in earlier step. The block diagram of PSF algorithm shown in Figure 1 is proposed by Martínez-Álvarez et al. (2008). The PSF algorithm is a closed loop process, hence it adds an advantage that it can attempt to predict the future values up to long duration by appending earlier forecasted value to existing original time series data. The block diagram for PSF algorithm shows the close loop feedback characteristics of the algorithm.

Figure 1: Block diagram of PSF algorithm methodology.

The clustering part consists of various tasks including data normalization, optimum cluster size selection and k-means clustering. The ultimate goal of this step is to make clusters of time series data and label them accordingly.

Normalization is one of the essential process in any time series data processing technique. Normalization is used to remove redundancies present in the data. The algorithm (Martinez Alvarez et al., 2011) used the following transformation to normalize the data.

$$Xj = \frac{Xj}{\frac{1}{N}\sum_{i=1}^{N} Xi} \tag{1}$$

where Xj is the input time series data and N is the total length of time series with units of time.

The reference articles (Martínez-Álvarez et al., 2008)(Martinez Alvarez et al., 2011) used the k-means clustering techniques to differentiate the data under different labels. The advantage of k-means clustering is its simplicity, but it requires the number of clusters as a reference value. In Martínez-Álvarez et al. (2008), the Silhouette index was used to decide the optimum numbers of clusters, whereas in improved version (Martinez Alvarez et al., 2011), three different indexes were used, which include Silhouette index (Kaufman and Rousseeuw, 2009), the Dunn index (Dunn†, 1974) and the Davies Bouldin index (Davies and Bouldin, 1979). But it is not necessary that the number of groups suggested by each of these indexes will be the same. Hence, it was suggested to select the cluster size which has been suggested by more than one index. After this, Jin et al. (2014) attempted to improve the basic PSF algorithm. They suggested that, instead of using multiple indexes, a single index could make computation simpler and used the Davies Bouldin index to obtain cluster size. The Silhouette index is based on distance of each object with all other objects in the clusters. It also

considers the dissimilarity of object with other clusters. The Silhouette index is given by:

$$silh(i) = \frac{a(i) - b(i)}{max(a(i), b(i))} \tag{2}$$

where, *a(i)* is the average distance of an object with other objects belonging to same cluster and *b(i)* is dissimilarity with the nearest neighbor cluster. Depending on *silh(i)* values, the possible number of groups are decided as discussed in Kaufman and Rousseeuw (2009). As an output of clustering technique, the original time series data get converted into series of labels. This series of labels is given as input to prediction block of the second phase of the PSF algorithm. The prediction technique consists of window size selection, searching of pattern sequences and estimation processes.

Let *x(i)* is the vector of time series data such that $x(i) = [x_1, x_2, x_3, x_4 \ldots\ldots x_n]$. After clustering and labeling, the vector converted to $y(i) = [L_1, L_2, L_3, L_2, L_1, L_1 \ldots..]$, where L_1, L_2.., are labels corresponding to the cluster centers to which data in vector *x(i)* belongs.

Then the searching process includes selection of last *W* labels in *y(i)* vector and it searches these labels all over in *y(i)*. If this sequence of last *W* labels did not repeat in the whole *y(i)* vector, then the search process repeats for last *(W - 1)* labels. In PSF algorithm, the width of this label sequence is named as '*Window size*'. The process of searching is get repeated itself till a window of sequence repeats itself more than once in vector *y(i)*. The worst case window size could be 1, since *y(i)* is a vector of clustered data, and hence every cluster label should repeat at least once in the whole list. The window size can vary from *W* to 1, but the selection of optimum value of window size is very critical and important to make more accurate prediction. The optimum window size selection is done in such a way that forecasting error will be minimum for training data set. Mathematically, the error function to be minimized is :

$$\sum_{d \epsilon TS} \left\| \overline{X}(d) - X(d) \right\| \tag{3}$$

where $\overline{X}(d)$ are predicted values and *X(d)* are original values of time series data. In practice, the window size selection is done with cross validation technique. All possible window sizes are tested on sample data and corresponding prediction errors are compared. The window size with minimum error considered as the optimum window size for prediction.

Once the optimum window size is obtained, the pattern sequences available in the window is get searched in label vector *y(i)* and the label present just next to each sequence is noted in a new vector *ES*. Finally, the future time series value is predicted by averaging the values in vector *ES* as given below:

$$\bar{X}(d) = \frac{1}{size(ES)} \times \sum_{j=1}^{size(ES)} ES(j) \tag{4}$$

where, *size(ES)* is the length of vector *ES*. The procedure of prediction in PSF algorithm is described in Figure 2.

Figure 2: Prediction with PSF algorithm.

The algorithm, till now, is in position to predict the future value for next interval of the time series. But, to make it applicable for long term prediction, the predicted short term values will get linked with original data and the whole procedure will be carried out till the desired length of time series prediction is obtained.

Martinez Alvarez et al. (2011) improved the label based forecasting (LBF) algorithm proposed in Martínez-Álvarez et al. (2008) to forecast the electricity price and compared it with earlier available forecasting algorithms like ANN (Catalão et al., 2007), ARIMA (Conejo et al., 2005), mixed models (García-Martos et al., 2007) and WNN (Lora et al., 2007). This comparison conclude that the PSF algorithm is able to outperform over all of these forecasting algorithms. Jin et al. (2014) proposed the limitations in PSF algorithm and suggested minute modification to minimize the computation delay. Majidpour et al. (2014) compared PSF algorithm with kNN and ARIMA, and observed that the PSF is worst performing algorithm in forecasting electric vehicle charging energy consumption. It also proposed few modifications in existing PSF algorithm. In modified PSF (MPSF), output is set such that it will get searched at the centre of window of sequence instead of at the end of window, as in original PSF. Koprinska et al. (2013) attempted to propose a new algorithm for electricity demand forecast, which is combination of PSF and Neural networks (NN) algorithms. The results concluded that PSF-NN is performing better than original PSF algorithm. Fujimoto and Hayashi (2012) modified the clustering method in PSF algorithm. It used a cluster method based on non-negative tensor factorization instead of k-means technique and forecasted energy demand using Photovoltaic energy

records. These literature show the need of the PSF algorithm in the field of time series forecasting that can be applied in any domain. Though, many of these articles concluded that PSF algorithm is able to outperform other time series forecasting algorithms, the PSF algorithm was unable to attract the attention of data analyst and researchers. Because, the PSF algorithm consists of many dependent functions and it is time consuming task to to decide optimum parameters to apply those functions for prediction. This **PSF** package aims to make PSF algorithm handy and quick with minimum efforts for coding.

Introduction to package PSF

This section is the introduction to the **PSF** package for R language. Discussing about dependency, this package imports package '**cluster**' (Maechler et al., 2015) and suggest packages '*knitr*' and '*rmarkdown*'. PSF package consists of various functions. These functions are designed such that these can replace the block diagrams of methodology used in PSF algorithm. Block diagram in Figure 3 represents the methodology mentioned in Figure 1 with the replacement of equivalent functions used in **PSF** package.

Figure 3: Block diagram with PSF package functions replacement for PSF algorithm.

This section discuss about each of these functions, their functionality along with simple examples. All functions in **PSF** package will be loaded with:

```
library(PSF)
```

Optimum cluster size selection:

As mentioned earlier, clustering of data is one of the initial phase of the PSF algorithm. The reference articles (Martínez-Álvarez et al., 2008) (Martinez Alvarez et al., 2011) have chosen the k-means clustering technique for generating data clusters according to the time series data characteristics. But the limitation of k-means clustering technique is that the suitable number of clusters are to be suggested by user. Hence, to avoid such situations, **PSF** package contains a function '*optimum_k( )*' which decides the optimum value of cluster size (k) with reference to the Silhouette index. This function generates the optimum cluster size as an output. In Martinez Alvarez et al. (2011), multiple indexes (Silhouette index, Dunn index and the Davies - Bouldin index) were considered to decide optimum cluster size. For the sake of simplicity and to save the calculation time, only the Silhouette index is considered in **PSF** package as suggested in Martínez-Álvarez et al. (2008).

The syntax of this function is: *optimum_k (data_in)*

This function takes '*data_in*' as input time series data, in any format be it matrix, data frame, list or vector. The content in input data should be strictly numeric. This function remove all *NA* values while initiating the process. This function returns the optimum value of cluster size (k) in numeric format. In R console, this function executes as shown below:

```
# Considering ‘data_in‘ = iris[1]
optimum_k(iris[1])

#> [1] 9
```

In this example, the function '*optimum_k( )*' suggests that a cluster size 9 will be the optimum value for grouping the first column of '*iris*' dataset.

Optimum Window size selection:

Once clustering of data is done, the optimum window size needs to be selected. This is an important but tedious and time consuming process, if it is done manually. In Martínez-Álvarez et al. (2008) and Martinez Alvarez et al. (2011), the selection of optimum window size is done through cross validation, in which data is partitioned in two subsets. One subset is reserved for analysis and other subset is for validating the analysis. Since, the window size will always be dependent on the pattern of the experimental data, it is necessary to determine the optimum window size for all time series data.

In **PSF** package, optimum window size selection is done with the function '*optimum_w( )*', which takes time series data and an integer '*next_val*' as input. This function suggests the optimum value of window size, such that the error between predicted and original data will be minimum. Internally,

this function divides the input time series data in two sub-parts. One of them is the last *'next_val'* numbers of data values which will be taken as reference to compare with the predicted values and to calculate RMSE values and other sub-part is the remaining data set which is used as training part of data set. The predicted values of window size with minimum RMSE value is taken as optimum window size. If more than one window size values are obtained with the same RMSE values, the maximum window size is preferred by the function *'optimum_w( )'*.

The syntax of function *'optimum_w( )'* is *optimum_w (data_in, next_val)*

As a result, this function returns the optimum window size (*w*), its corresponding RMSE value and predicted values with corresponding to each window size as explained with an example:

```
# Considering ‘data_in‘ = iris[1] and ‘next_val‘ = 5
optimum_w(iris[1], 5)

#> $Optimum_W
#> [1] 1
#>
#> $RMSE_Values
#> [1] 0.3059953
#>
#> $Prediction
#>  Original_Data   W_size.1
#> 1           6.7         6.461538
#> 2           6.3         6.461538
#> 3           6.5         6.461538
#> 4           6.2         6.461538
#> 5           5.9         6.461538
```

Each column in the table is time series data. First column represents data under testing from the original data set and all other column presents the predicted data for various window sizes varies from 1 to next possible value of window size. The maximum value of W is considered till the sequence of last *W* numbers doesn't repeat itself in particular dataset.

Prediction with PSF:

In PSF algorithm, prediction procedure is succeeded by data processing, optimum window size and cluster size selection. In **PSF** package, the prediction is done with function *'pred_for_w( )'*, which takes time series data, window size (w), cluster size (*k*) and an integer *'next_val'*. Value *'next_val'* indicates the count up to what extend the forecasting is to be done by function *'pred_for_w( )'*. The time series data taken as input can be in any format supported by R language. The **PSF** package automatically convert it in suitable format and proceeds further. Whereas other input parameters *w*, *k* and *next_val* should be in integer format.

The function *'pred_for_w( )'* initiates with process of normalization of data with function *'data_norm( )'*. Then the function *'pred_for_w( )'* selects the last *'w'* numbers of values in input time series data and searche that number sequence in the whole dataset. Along with this, it captures the very next integer value after each sequence and calculates average of these values. The obtained averaged value along with denormalization is considered as raw predicted value. If the input parameter *'next_val'* is greater than unity, then the predicted value is appended to the original input data and repeats the procedure *'next_val'* times. Finally, the processed data is denormalized with function *'data_denorm( )'* and returns a time series which replace the labels with their equivalent cluster centers.

The PSF package contains another function *pred_for_w_plot()*, which is almost similar to *pred_for_w()* except in its capability to plot the graph showing input and predicted time series data. The syntax of this function is *pred_for_w (data_in, w, k, next_val)*. The example below shows the applicability of this function in forecasting. The plot obtained in following example is shown in Figure 4.

```
# Considering ‘data_in‘ = iris[1], ‘w‘ = 3, ‘k‘ = 4 and ‘next_val‘ = 5
pred_for_w(iris[1], 3, 4, 5)

#> [1] 7.082143 6.271795 5.623684 5.623684 5.623684

# Considering ‘data_in‘ = iris[1], ‘w‘ = 3, ‘k‘ = 4 and ‘next_val‘ = 5
pred_for_w_plot(iris[1], 3, 4, 5)
```

```
#> $Predicted_Values
#> [1] 7.082143 6.271795 5.623684 5.623684 5.623684
#>
#> $Plot
```

Figure 4: Graph showing forecasted results with function '*pred_for_w_plot( )*'

Along with this, the **PSF** package is also contains '*AUTO_PSF( )*' function to automates all other functions present in the package. This function takes time series data and *next_val* integer as input similar to '*pred_for_w( )*' and automatically do all operations like data normalization, optimum *k* value selection, optimum window size selection, prediction and data denormalization.

The syntax of function '*AUTO_PSF( )*' is *AUTO_PSF (data_in, next_val)*.

As an output, '*AUTO_PSF( )*' generate the predicted data series containing *'next_val*' numbers. Along with this, it generates the plot in Figure 5, similar to function '*pred_for_w_plot( )*' as shown in example below.

```
# Considering 'data_in' = iris[1] and next_val = 3
AUTO_PSF(iris[1],3)

#> $Predicted_Values
#> [1] 5.142857 5.966667 5.142857
#>
#> $Plot
```

Figure 5: Graph showing forecasted results with function '*AUTO_PSF( )*'.

Other Supportive Functions:

Apart from above mentioned functions, the package **PSF** consists of other supportive functions, which include functions *data_norm( )*, *data_denorm( )*, *rmse( )*, *mae( )* and *plot_PSF( )*.

The *data_norm( )*/*data_denorm( )* functions:

Function '*data_norm( )*' takes time series data in any of the data format supported by R language and returns the normalized form of input data along with maximum and minimum values obtained in input data series in order to assist the process of denormalization at the end of process of forecasting.

In **PSF** package, the process of denormalization is done with function '*data_denorm( )*', which takes the time series data, maximum and minimum values returned by function '*data_norm( )*' as input parameters and generates the denormalized data series. Both of these functions are used in functions '*pred_for_w( )/pred_for_w_plot( )*', where the input time series data in normalized initially and the predicted data by PSF algorithm is denormalized before displaying it as output. The working of these functions are explained below with an example, where variable '*a*' is the input vector with 10 entries and variable '*b*' is the output of function '*data_norm( )*'. Along with normalized data series, this function returns maximum and minimum numbers available in the vector variable '*a*'. These values will be referred in the function '*data_denorm( )*'. Here, variable '*c*' is the output of function '*data_denorm( )*', which takes vector '*b*' and other parameters returned by function '*data_norm( )*' and regenerates the data series similar to original data vector '*a*'.

```
a <- c(0,1,2,3,4,5,6,7,8,9)
b <- data_norm(a)
b

#> $x
#> [1] 0.0000000 0.1111111 0.2222222 0.3333333 0.4444444 0.5555556 0.6666667
#> [8] 0.7777778 0.8888889 1.0000000
#>
#> $dmax
#> [1] 9
```

```
#>
#> $dmin
#> [1] 0

c <- data_denorm(b$x, b$dmax, b$dmin)
c

#> [1] 0 1 2 3 4 5 6 7 8 9
```

The *rmse( )/mae( )* functions:

In PSF algorithm, RMSE and MAE values were compared to check how accurate prediction is done by the algorithm. The functions '*rmse( )*' and '*mae( )*' are used in **PSF** package to produce RMSE and MAE values, respectively. Both of these functions consumes the difference between original time series data and corresponding predicted time series data, and calculates RMSE and MAE values as shown in next example.

```
## Generate 100 random numbers within some limits
x <- sample(1:7, 100, replace = TRUE)
y <- sample(1:4, 100, replace = TRUE)

z1 <- rmse(x - y)
z1

#> [1] 2.590367

z2 <- mae(x - y)
z2

#> [1] 2.13
```

The *plot_PSF( )* function:

All plot related operations in **PSF** package are done with function '*plot_PSF( )*'. This function takes original time series data and predicted data as input and plot the corresponding graph plot. In the plot, red colored region indicates original data and blue colored region represents predicted data. The plot obtained by '*plot_PSF( )*' possesses dynamic margin size such that it can include input dataset and all predicted values as per requirements.

```
## Generate x as present data of 100 random numbers within some limits
x <- sample(1:7, 100, replace = TRUE)
## Generate y as forecasted data of 20 random numbers
y <- sample(1:4, 20, replace = TRUE)
plot_PSF(x,y)
```

Figure 6: Graph showing forecasted results with function '*plot_PSF( )*'.

Example

This section discusses about the example to introduce use of **PSF** package and to compare it with *auto.arima()* which is one of the well accepted R package used for forecasting. The data used in this example is a time series with continuous and symmetrical pattern, such that human being can easily predict the future values without using any computational tool. Consider a ramp function with amplitude of 5 as time series signal as shown in Figure 7. First of all, the function '*AUTO_PSF( )*' from **PSF** package is used to forecast future values. Figure 8 shows the prediction plot with function '*AUTO_PSF( )*'. It is observed that, the predicted values follows the same pattern as of original data.

Figure 7: Time series data with symmetrical pattern.

```
library(PSF)
AUTO_PSF(data_in,5)

#> $Predicted_Values
#> [1] 1 2 3 4 5
```

Figure 8: Prediction plot as an output of function '*AUTO_PSF( )*'.

Now, consider '*auto.arima( )*' function from R package '**forecast**'. This function compares either AIC, AICc or BIC values and suggest best ARIMA model for the given time series data (Hyndman et al., 2015). Figure 9 shows the prediction plot obtained with this function.

```
library(forecast)
forecast(auto.arima(data_in))

#> Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
#> 21              3 1.187612 4.812388 0.2281924 5.771808
#> 22              3 1.187612 4.812388 0.2281924 5.771808
#> 23              3 1.187612 4.812388 0.2281924 5.771808
#> 24              3 1.187612 4.812388 0.2281924 5.771808
#> 25              3 1.187612 4.812388 0.2281924 5.771808
#> 26              3 1.187612 4.812388 0.2281924 5.771808
#> 27              3 1.187612 4.812388 0.2281924 5.771808
#> 28              3 1.187612 4.812388 0.2281924 5.771808
#> 29              3 1.187612 4.812388 0.2281924 5.771808
#> 30              3 1.187612 4.812388 0.2281924 5.771808

plot(forecast(auto.arima(data_in)))
```

Figure 9: Prediction plot as an output of function '*auto.arima( )*'.

From Figures 8 and 9, it can be observed that for given time series '*data_in*', the output with **PSF** package is much reasonable compared to other. Surely, this example is not enough to prove that PSF algorithm outperforms the ARIMA model suggested by *auto.arima( )* function, since various other parameters for model fitting may not be considered, but it certainly indicates the usability of the **PSF** package for better univariate time series predictions.

Conclusions

This article is about brief summary about the R package, **PSF**. This package is introduced to promote the algorithm Pattern Sequence based Forecasting (PSF) which was proposed by Martínez-Álvarez et al. (2008) and then modified and suggested improvement by Martinez Alvarez et al. (2011). The functions involved in **PSF** package can be tested with a time series data in any format like vector, list or matrix. The aim of the **PSF** package is to simplify the calculations and to automate the steps involved in prediction while using PSF algorithm.

Acknowledgements

I would like my colleague, Mr. Sudhir Kumar Mishra, for his valuable comments and suggestions to improve this article. Thanks also goes to CRAN maintainers for needful comments on the submitted package.

Bibliography

J. Catalão, S. Mariano, V. Mendes, and L. Ferreira. Short-term electricity prices forecasting in a competitive market: a neural network approach. *Electric Power Systems Research*, 77(10):1297–1304, 2007. [p]

A. J. Conejo, M. A. Plazas, R. Espinola, and A. B. Molina. Day-ahead electricity price forecasting using the wavelet transform and arima models. *Power Systems, IEEE Transactions on*, 20(2):1035–1042, 2005. [p]

D. L. Davies and D. W. Bouldin. A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (2):224–227, 1979. [p]

J. C. Dunn†. Well-separated clusters and optimal fuzzy partitions. *Journal of cybernetics*, 4(1):95–104, 1974. [p]

Y. Fujimoto and Y. Hayashi. Pattern sequence-based energy demand forecast using photovoltaic energy records. In *Renewable Energy Research and Applications (ICRERA), 2012 International Conference on*, pages 1–6. IEEE, 2012. [p]

C. García-Martos, J. Rodríguez, and M. J. Sánchez. Mixed models for short-run forecasting of electricity prices: application for the spanish market. *Power Systems, IEEE Transactions on*, 22(2):544–552, 2007. [p]

R. J. Hyndman, C. Cinelli, Y. Khan, Z. Mayer, S. Razbash, D. Schmidt, D. Shaub, Y. Tang, E. Wang, Z. Zhou, et al. Package 'forecast', 2015. [p]

C. H. Jin, G. Pok, H.-W. Park, and K. H. Ryu. Improved pattern sequence-based forecasting method for electricity load. *IEEJ Transactions on Electrical and Electronic Engineering*, 9(6):670–674, 2014. [p]

L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009. [p]

I. Koprinska, M. Rana, A. Troncoso, and F. Martínez-Álvarez. Combining pattern sequence similarity with neural networks for forecasting electricity demand time series. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–8. IEEE, 2013. [p]

A. T. Lora, J. M. R. Santos, A. G. Exposito, J. L. M. Ramos, and J. C. R. Santos. Electricity market price forecasting based on weighted nearest neighbors techniques. *Power Systems, IEEE Transactions on*, 22(3):1294–1301, 2007. [p]

M. Maechler, P. Rousseeuw, A. Struyf, M. Hubert, K. Hornik, M. Studer, and P. Roudier. Package 'cluster', 2015. [p]

M. Majidpour, C. Qiu, P. Chu, R. Gadh, and H. R. Pota. Modified pattern sequence-based forecasting for electric vehicle charging stations. In *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*, pages 710–715. IEEE, 2014. [p]

F. Martínez-Álvarez, A. Troncoso, J. C. Riquelme, and J. S. A. Ruiz. Lbf: A labeled-based forecasting algorithm and its application to electricity price time series. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 453–461. IEEE, 2008. [p]

F. Martinez Alvarez, A. Troncoso, J. C. Riquelme, and J. S. Aguilar Ruiz. Energy time series forecasting based on pattern sequence similarity. *Knowledge and Data Engineering, IEEE Transactions on*, 23(8): 1230–1243, 2011. [p]

Neeraj Bokde
Visvesvaraya National Institute of Technology, Nagpur
North Ambazari Road, Nagpur
India
neerajdhanraj@gmail.com

Kishore Kulat
Visvesvaraya National Institute of Technology, Nagpur
North Ambazari Road, Nagpur
India
kdkulat@ece.vnit.ac.in

This figure "Figure_1.png" is available in "png" format from:

http://arxiv.org/ps/1606.05492v1

This figure "Figure_2.png" is available in "png" format from:

http://arxiv.org/ps/1606.05492v1

This figure "Figure_3.png" is available in "png" format from:

http://arxiv.org/ps/1606.05492v1

This figure "Figure_4.png" is available in "png" format from:

http://arxiv.org/ps/1606.05492v1

This figure "Figure_5.png" is available in "png" format from:

http://arxiv.org/ps/1606.05492v1

This figure "Figure_6.png" is available in "png" format from:

http://arxiv.org/ps/1606.05492v1

This figure "Figure_7.png" is available in "png" format from:

http://arxiv.org/ps/1606.05492v1

This figure "Figure_8.png" is available in "png" format from:

http://arxiv.org/ps/1606.05492v1

This figure "Figure_9.png" is available in "png" format from:

http://arxiv.org/ps/1606.05492v1