An Assignment Report

on

Implementation of Chaotic Analysis on Retweet Time Series

Submitted by

Team Number 5 :
Ohileshwar Itagi - 13IT251
Ashish Singh - 13IT107
Shyam BS - 13IT144
VII Sem B.Tech (IT)

in partial fulfillment for the award of the degree
of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

At

Department of Information Technology
National Institute of Technology Karnataka, Surathkal.
November 2016

CERTIFICATE

This is to certify that the assignment entitled ”Implementation of Chaotic Analysis on Retweet Time Series” is a bonafide work carried out under my guidance for the course IT367, by Ohileshwar Itagi(13IT251), Shyam B S(13IT144), Ashish Singh(13IT107) students of VII Sem B.Tech (IT) at the Department of Information Technology, National Institute of Technology Karnataka, Surathkal, during the academic year 2016-17, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Information Technology, at NITK Surathkal.

Place: Surathkal

(Signature of the Guide)

Date: 28-11-2016

Name of Guide: Mr. Biju R Mohan

Name and Address of Organization: NITK, Surathkal

DECLARATION

I hereby declare that the assignment entitled ”Implementation of Chaotic Analysis on Retweet Time Series”, submitted by us for the course IT361 as part of the partial course requirements for the award of the degree of Bachelor of Technology in Information Technology at NITK Surathkal is my original work. I declare that the project has not formed the basis for the award of any degree, associateship, fellowship or any other similar titles elsewhere.

(Signature of the Students)

Place: Surathkal
Date: 28-11-2016

Abstract

This report presents a technique of performing a 0-1 Chaos Test on the Retweet Time Series and applying a LS-SVM (Least Square Support Vector Machine) model in R. It further evaluates our LS-SVM model based on the goodness of fit measure called MAPE(Mean Absolute Percentage Error). We obtain 10 weeks of data from the standard Siena Weibo(Chinese Twitter) data set, which is freely available online. We pre-process the data set to obtain the Retweet Time Series and perform 0-1 Chaos test on it. Once we collect all the time series which are chaotic in nature we apply an LS-SVM model to each time series. For the LS-SVM model we use 70% for training the data and 30% as the test data. Once we get the predicted values of the retweet counts, we compare them with the actual values and compute the MAPE(Mean Absolute Percentage Error) of our model.

Contents

List of Figures

List of Tables

1 Introduction

We have downloaded 10 weeks of Chinese Twitter data and extracted a total of 300 tweets(called as Root Tweets) which have the highest Retweet count. We divide this report mainly into 2 parts. The first part will contain the procedure to execute the code and explanation of each line of the code, when the analysis is carried out for a single Time Series (Retweet Time series of a single Root Tweet). The second part will contain the procedure to execute the code and explanation of each line of the code, when the analysis is carried out for all the Time Series (Retweet Time series of every Root Tweet).

2 Methodology

2.1 Development Environment

The R packages required to run our code are as follows:

1. Chaos01 :- This R package allows us to perform 0-1 test for any given time series.
2. minpack.lm :- This R package allows us to apply LS-SVM(Least Square Support Vector Machines) model on any given time series.
3. nonlinearTseries :- This R package contains a function called as "timeLag()" which allows us to compute the time delay τ of the time series.
4. tseriesChaos :- This R package contains an "embedd()" function which is used to perform the phase space reconstruction of the Chaotic Retweet Time Series.
5. fractal :- This R package contains the function "corrDim()" which is used to plot and find the embedding dimension of the time series.

All these packages are available in CRAN project.

2.2 Procedure for single Retweet Time Series

In the project folder which we shared, navigate to the "Single Time Series" directory. There are 3 files in the directory. They are as follows:-

1. 20338_mkpfcBeWtW.csv :- The csv file containing the Retweet Time Series of length 20338 and the Root tweet id being "mkpfcBeWtW".
2. chaosTest.R :- This file contains the R code to perform the 0-1 test to the time series to determine whether the time series is chaotic or not. If the output of the 0-1 test is 1 then the time series is chaotic in nature. If the output is 0 then the time series is not chaotic.
3. hopefully_last_2.R :- This file contains the R code to fit the Retweet Time Series using a LS-SVM (Least Square Support Vector Machine) model. Once the predictions are made, it also contains the code to compute MAPE(Mean Absolute Percentage Error). MAPE is one of the standard goodness of fit measures.

First we perform the 0-1 test for Chaos on our Retweet Time Series. Next we apply the LS-SVM model on our Chaotic Retweet Time Series.

2.2.1 R code for 0-1 Chaos Test on Single Retweet Time Series

The below code is present in the file ”chaosTest.R”.

1.
```r
#To install the necessary packages
packages_list =
    c("Chaos01","minpack.lm","nonlinearTseries","tseriesChaos", "fractal")
for (package in packages_list) {
  if (!require(package, character.only=T, quietly=T)) {
    install.packages(package)
    library(package, character.only=T)
  }
}
```

   This line of code checks if all the packages required to run the project are present or not. If they are not present then it will try to install those packages.

2.
```r
seriesFile <- read.csv(file="20338_mkpfcBeWtW.csv",head=FALSE,sep=",")
```

   This line reads the ”20338_mkpfcBeWtW.csv” file which contains the Retweet Time Series data into the variable called as ”seriesFile” where the separator is ”,” and it also reads the headers of the columns. There is no need to specify the absolute path of the Retweet Time Series File in the file parameter, But make sure that the file has got read permissions.

3.
```r
series <- seriesFile$V1
```

   This line is used to extract the first column from the csv file which we just read

4.
```r
res <- testChaos01(series,out=TRUE,c.int=c(0,2*pi),c.rep=1000)
```

   This line is used to compute the average K_c and plot the K_c of the Retweet Time Series versus various values of c. The plot is shown in figure 4. The K_c value measures the Asymptotic Growth Rate of the Mean Square Displacement of a time series. c.int=c(0,2π) and c.rep=1000 indicates that the c value was varied between (0,2π) 1000 times and the average of those K_c values was computed. The res variable holds the K_c versus c plot and also the average K_c value.

5.
```r
plot(res)
```

   This line is used to plot the K_c versus c plot of the Asymptotic Growth Rate vs various values of c between 0 to 2π. The plot is shown in the Figure 1. As you can see from the figure the time series which we have chosen has K_c value as 1, which means it is completely chaotic.

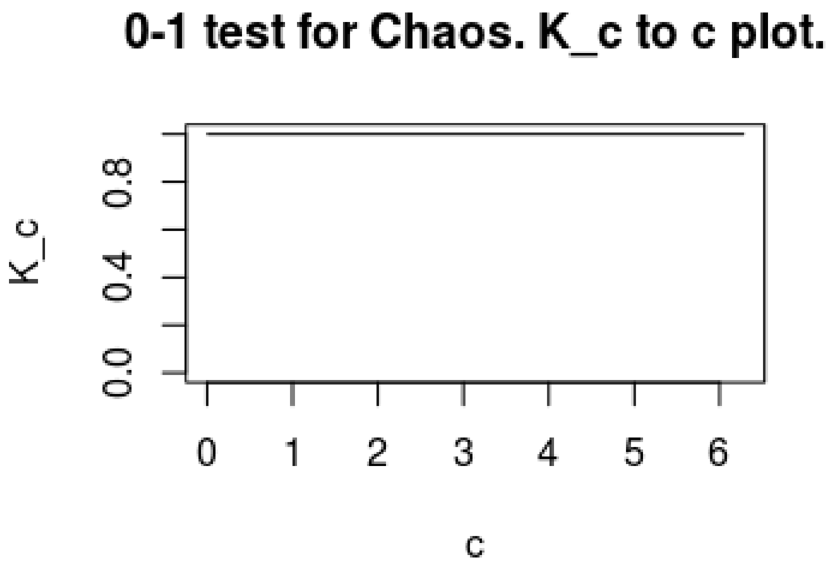

Figure 1: K_c vs c plot.

6.
```
res2 <- testChaos01(series,out=FALSE,c.int=c(0,2*pi),c.rep=1000)
```

This line is different from the third line in the parameter "out=TRUE". Here, out="False" means res2 just holds the average K_c and doesnt contain the plot.

7.
```
print(res2)
```

This line prints the average K_c for the time series present in "20338_mkpfcBeWtW.csv". If the average K_c value is greater than 0.7 (chaotic threshold) then we consider that particular Retweet Time Series to be chaotic in nature.

2.2.2 R code for fitting a LS-SVM model on Single Retweet Time Series

The below code is present in the file "hopefully_last_2.R".

1.
```r
#To install the necessary packages
packages_list =
    c("Chaos01","minpack.lm","nonlinearTseries","tseriesChaos", "fractal")
for (package in packages_list) {
  if (!require(package, character.only=T, quietly=T)) {
    install.packages(package)
    library(package, character.only=T)
  }
}
```

   This line of code checks if all the packages required to run the project are present or not. If they are not present then it will try to install those packages.

2.
```r
seriesFile <- read.csv("20338_mkpfcBeWtW.csv",head=FALSE,sep=",")
```

   This line reads the "20338_mkpfcBeWtW.csv" file which contains the Chaotic Retweet Time Series data into the variable called as seriesFile where the separator is "," and it also reads the headers of the columns. Please specify the absolute path of the Chaotic Retweet Time Series File in the file parameter and make sure that the file has got read permissions.

3.
```r
series <- seriesFile$V1
original_series <- series
```

   This section is used to extract the first column from the csv file which we just read. Copy the entire series to original_series because the "series" variable is expected to be updated with only 70% data for training purposes.

4.
```r
series <- series[1:(n*0.7)]
n=length(series)
```

   This section is first used to update the series variable to 70% of the data. Then it is used to extract the length of the chaotic time series and store it in a variable named "n".

   We need to compute the time delay "τ" and the embedding dimension "m".

5.
```r
max_tau=400
```

```
tau=nonlinearTseries::timeLag(series, technique = "ami",lag.max =
    max_tau, do.plot = TRUE,selection.method="first.minimum")
```

The first line is used to assign the largest value that τ can possibly take. To compute τ we use the timeLag function in the nonLinearTseries package. The parameters we pass are as follows. It uses the Average Mutual Information("ami") method to compute τ. The largest value of τ can be "max_tau" as defined earlier. do.plot=TRUE indicates that the plot of average mutual information versus time graph must be drawn as shown in the Figure 2. The time delay value will be equal to the first minimum value in the plot as specified in our paper.

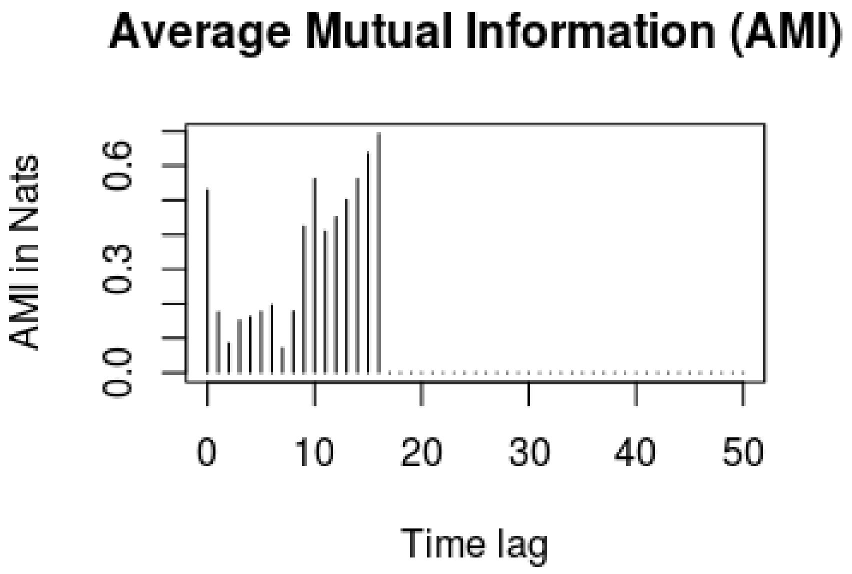

Figure 2: AMI vs time lag

6.
```
emb_dim=fractal::corrDim(series, dimension=7, olag=0, resolution=2)
plot(emb_dim)
m=emb_dim=3
```

This method is used to compute the embedding dimension of the time series. It takes in the timeSeries, the largest dimension, olag and resolution as it's parameters. The second line plots graphs for various embedding dimension values as shown in the Figure 3. From the graph which we plotted we come to know that the best value for embedding dimension is 3. The paper also suggests that the embedding dimension value must be equal to 3. Any other value leads to lesser accuracy in our predictions. Thus this line is used to set the value of embedding dimension "m" to be equal to 3.

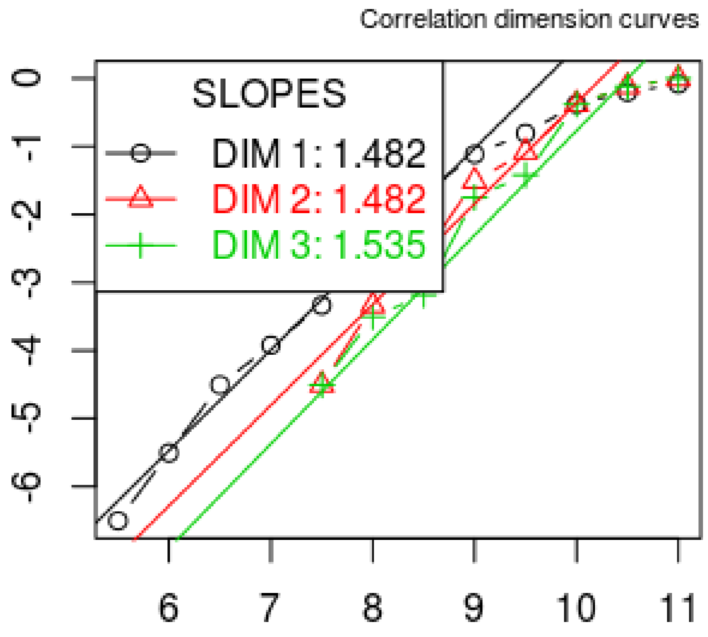

Figure 3: lnC(r) vs ln(r)

We now construct a Phase Space Re-Construction of our Chaotic Retweet Time Series.

7.
```
matrix=tseriesChaos::embedd(series,d=tau,m=emb_dim)
```

This line performs the phase space reconstruction to our Chaotic Retweet Time Series. The Time Series is changed from single dimension to multidimensional("m" dimensional) time series.

The time series with the new phase is then fitted with a LS-SVM model.

8.
```
x=matrix[1:(nrow(matrix)-1),]
y=matrix[2:(nrow(matrix)),]
```

This section assigns x,y to be the multidimensional training input,output to the SVM model respectively.

9.
```
x <- data.frame(x)
y <- data.frame(y)
```

This section assigns x,y from their matrix values to their data frame values respectively.

10.
```
columnNames=vector(mode="list",length=m+1)
columnNames[1]="y"
for(i in 2:(m+1)){
  columnNames[i]=paste("x",i-1,sep="")
}
```

This section is used to generate the column names(y, x1, x2 ..) which is attached to the data frames produced. This naming is necessary to train the svm appropriately.

11.
```
#train all the models
models=vector(mode="list",length=m)
for(i in 1:m){
  y_ <- y[,i]
  x_y <- cbind(y_,x)
  colnames(x_y) <- columnNames
  models[[i]] <- lm(y ~ . , x_y)
}
```

We use multiple regression for every value in the Yi vector to be predicted. Hence if the Yi vector contains m values, we will build m svm models. The last line of code in the above snippet, trains a particular svm model by feeding the built data frame accordingly to the svm.

12.
```
M=nrow(x)
```

This line stores the length of the multidimensional x in the variable M.

13.
```
X_now <- data.frame(x[1,])
colnames(X_now) <- columnNames[2:(m+1)]
```

After training is done, for testing purpose, we have to predict the entire time series for from the first Xi vector. In this section the first vector X1 is initialized.

14.
```
predicted_time_series=vector(mode="list", length=n)
predicted_time_series[1]=X_now[1,1]
```

The predicted_time_series vector is initialized.

15.
```
for(i in 2:n){
  X_next=X_now
  for(j in 1:m){
    X_next[1,j] = predict(models[[j]],newdata=X_now)
  }
  predicted_time_series[i]=X_next[1,1]
  X_now=X_next
}
```

Use the X_now vector (initially it is X1), to predict the Xi+1 vector using the svm models created. The first value in Xi+1'th vector will be the i+1'th value in time series. Store that value in the predicted_time_series vector.

16.
```
retweet_count = cbind(series, predicted_time_series)
matplot(retweet_count,pch=c(1,2),col=c("green","red"), ylim=c(0,10000))
```

This section of code binds together two time series together namely series and predicted_time_series. It then plots the two time series together on the same graph to indicate the predicted value and the original value as shown in the Figure 4.

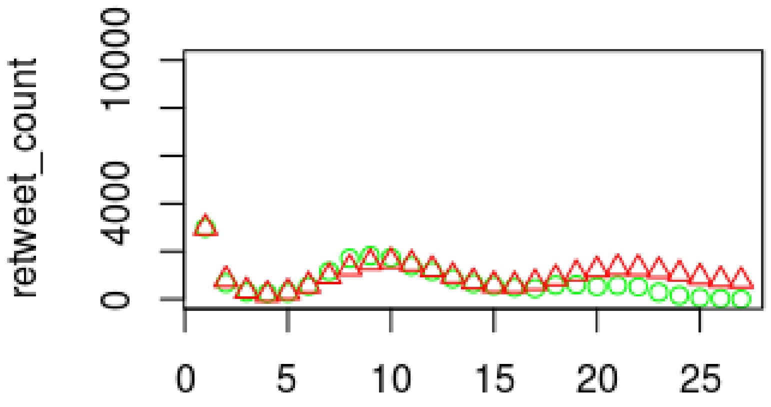

Figure 4: Green - actual series, Red - predicted series.

17.

```
print("MAPE : ")
MAPE=0
for(i in 1:n){
  if(series[i] > 0){
    MAPE=MAPE+abs(predicted_time_series[[i]] - series[i])/series[i]
  }
}
MAPE=MAPE/n
print(MAPE)
```

Once we have predicted the time series values we must compute the MAPE(Mean Absolute Percentage Error) of the LS-SVM model. The above section of code does exactly that.

2.3 Procedure for Multiple Retweet Time Series

In the project folder which we shared, navigate to the "Multiple Time Series" directory.

There are 2 files in the directory. They are as follows:-

1. chaosTest_allFiles.R :- This file contains the R code to perform the 0-1 test to the time series to determine whether the time series is chaotic or not. If the output of the 0-1 test is 1 then the time series is chaotic in nature. If the output is 0 then the time series is not chaotic. This is done for all files.

2. last_with_trainingset_reduced.R :- This file contains the R code to fit the Retweet Time Series using a LS-SVM (Least Square Support Vector Machine) model. Once the predictions are made, it also contains the code to compute MAPE(Mean Absolute Percentage Error). MAPE is one of the standard goodness of fit measures. This is done for all files.

There are 2 directories in the "Multiple Time Series" directory. They are as follows :

1. chaoticSeriesFiles : It contains the Retweet Time Series which are chaotic in nature. We create these files after executing the "chaosTest_allFiles.R" .

2. timeSeriesFiles_60 : It contains the retweet time series of all the root tweets. The files present in this directory are named as "sizeofTimeSeries_rootID.csv".

2.3.1 R code for 0-1 Chaos Test on Multiple Retweet Time Series

The below code is present in the file "chaosTest_allFiles.R" :

1.
```r
#To install the necessary packages
packages_list =
   c("Chaos01","minpack.lm","nonlinearTseries","tseriesChaos", "fractal")
for (package in packages_list) {
  if (!require(package, character.only=T, quietly=T)) {
    install.packages(package)
    library(package, character.only=T)
  }
}
```

   This line of code checks if all the packages required to run the project are present or not. If they are not present then it will try to install those packages.

2.
```r
this.dir <- dirname(parent.frame(2)$ofile)
setwd(this.dir)
fileNames <- list.files(paste0(getwd(),"/timeSeriesFiles_60"),
   pattern="*.csv", full.names=TRUE)
```

   This line reads all csv files which contains the Retweet Time Series data into the variable called as fileNames where the pattern is "*.csv" and it also reads the headers of the columns. There is no need to specify the absolute path of the Multiple Retweet Time Series Directory. Only make sure that the directory has got read permissions.

3.
```r
n=length(fileNames)
```

   This calculates the number of files in the directory.

4.
```r
Kc_values=numeric(0)
```

   This initialises an empty list of 0 length.

5.
```r
chaotic_threshold=0.7
```

   Set the threshold value to 0.7. If K_c less than 0.7, it will not be considered chaotic.

6.
```r
for(i in 1:n){
```

   Iterate over all the n individual retweet time series files.

7.
```
seriesFile <- read.csv(file=fileNames[i],head=FALSE,sep=",")
```

This line reads the current csv file which contains the Retweet Time Series data into the variable called as seriesFile where the separator is ”,” and it also reads the headers of the columns.

8.
```
series <- seriesFile$V1
```

This line is used to extract the first column from the csv file which we just read

9.
```
res1 <- testChaos01(series,out=TRUE,c.int=c(0,2*pi),c.rep=1000)
```

This line is used to compute the average K_c and plot the K_c of the Retweet Time Series versus various values of c. The plot is shown in figure 5. The K_c value measures the Asymptotic Growth Rate of the Mean Square Displacement of a time series. c.int=c(0,2π) and c.rep=1000 indicates that the c value was varied between (0,2π) 1000 times and the average of those K_c values was computed. The res1 variable holds the K_c versus c plot and also the average K_c value.

10.
```
res2 <- testChaos01(series,out=FALSE,c.int=c(0,2*pi),c.rep=1000)
```

This line is different from the third line in the parameter ”out=TRUE”. Here, out=”False” means res2 just holds the average K_c and doesnt contain the plot.

11.
```
if(is.na(res2) != TRUE){
```

Dont process the loop if res2 is not applicable.

12.
```
Kc_values <- c(Kc_values,res2)
```

Append the result, i.e the average K_c to the list of K_c values.

13.
```
print(res2)
```

This line prints the average K_c for the current time series.

14.
```
plot(res1, ylim = c(0,1))
```

This line is used to plot the K_c versus c plot of the Asymptotic Growth Rate vs various values of c between 0 to 2π.

15.
```
if(res2 > chaotic_threshold){
```

If the average K_c value is greater than 0.7 (chaotic threshold) then we consider that particular Retweet Time Series to be chaotic in nature.

16.
```
count=count+1
```

Increment the count of chaotic time series.

17.
```
file.copy(from = fileNames[i],paste0(getwd(),"/chaoticSeriesFiles"),
    copy.mode=TRUE)
```

Copy the file to the "chaoticSeriesFiles" directory if the time series is chaotic in nature.

18.
```
hist(Kc_values, breaks = 10, col = "blue")
```

Plot the histogram for all the average K_c values of all the retweet time series with breaks of 10 units. This plot gives an idea of how many retweet time series have a higher chaotic threshold. The plot which we obtained is shown in the Figure 5.

19.
```
print(paste0("Total number of time series processed : ",n))
print(paste0("Number of chaotic series (for threshold
    ",chaotic_threshold,") : ",count))
print(paste0("Percentage of chaotic series : ",count/n))
```

This prints the total time series processed, number of chaotic series, total percentage of retweet time series which are chaotic.

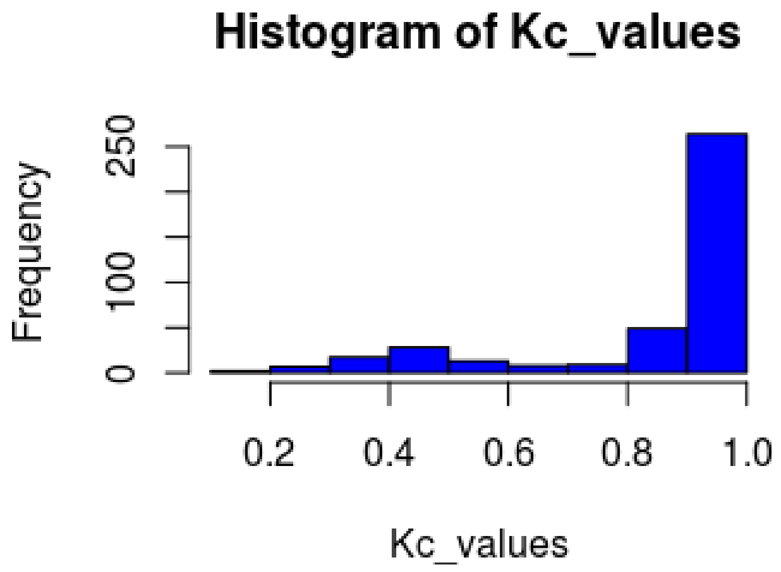

Figure 5: Retweet count vs Average Kc_value

2.3.2 R code for fitting a LS-SVM model on Multiple Retweet Time Series

The below code is present in the file ”last_with_trainingset_reduced.R”.

1.
```
#To install the necessary packages
packages_list =
    c("Chaos01","minpack.lm","nonlinearTseries","tseriesChaos", "fractal")
for (package in packages_list) {
  if (!require(package, character.only=T, quietly=T)) {
    install.packages(package)
    library(package, character.only=T)
  }
}
```

   This line of code checks if all the packages required to run the project are present or not. If they are not present then it will try to install those packages.

2.
```
this.dir <- dirname(parent.frame(2)$ofile)
setwd(this.dir)
```

   This line of code sets the current directory as the working directory.

3.
```
regressOn <- function(fileName){
```

   This creates a ’regressOn’ funtion which takes the fileName as a parameter for performing LS-SVM regression on the retweet time series.

4.
```
seriesFile <- read.csv(fileName,head=FALSE,sep=",")
```

   This line reads the file which contains the Chaotic Retweet Time Series data into the variable called as seriesFile where the separator is ”,” and it also reads the headers of the columns.

5.
```
series <- seriesFile$V1
```

   This line is used to extract the first column from the csv file which we just read.

6.
```
original_series <- series
```

   Copy the series into original_series variable.

7.
```
series <- series[1:(n*0.7)]
```

Convert the 'series' variable to the top 70 percent of the retweet time series. This will be used for training our model.

8.
```
n=length(series)
```

This line is used to extract the length of the chaotic time series and store it in a variable named "n". We need to compute the time delay "τ" and the embedding dimension "m".

9.
```
max_tau=400
```

This line is used to assign the largest value that τ can possibly take.

10.
```
tau=nonlinearTseries::timeLag(series, technique = "ami",lag.max =
   max_tau, do.plot = TRUE,selection.method="first.minimum")
```

To compute τ we use the timeLag function in the nonLinearTseries package. The parameters we pass are as follows. It uses the Average Mutual Information("ami") method to compute τ. The largest value of τ can be "max_tau" as defined earlier. do.plot=TRUE indicates that the plot of average mutual information versus time graph must be drawn as shown in the figure. The time delay value will be equal to the first minimum value in the plot as specified in our paper.

11.
```
emb_dim=3
m=emb_dim
```

Assign embedding dimension equal to 3 as specified in the paper. Variable 'm' is used to represent it.

12.
```
matrix=tseriesChaos::embedd(series,d=tau,m=emb_dim)
```

This line performs the phase space reconstruction to our Chaotic Retweet Time Series.

13.
```
x=matrix[1:(nrow(matrix)-1),]
y=matrix[2:(nrow(matrix)),]
```

Matrix 'x' consists retweet time series values from first to second last element of the file. Matix 'y' consists retweet time series values from second to last element of the file.

14.
```
x <- data.frame(x)
y <- data.frame(y)
```

Convert x and y matrices to dataframes for further processing.

15.
```
columnNames=vector(mode="list",length=m+1)
columnNames[1]="y"
```

Create a new matrix having a column named 'y'.

16.
```
for(i in 2:(m+1)){
   columnNames[i]=paste("x",i-1,sep="")
}
```

Copy data frame 'x' into columnNames.

17.
```
models=vector(mode="list",length=m)
for(i in 1:m){
  y_ <- y[,i]
  x_y <- cbind(y_,x)
  colnames(x_y) <- columnNames
  models[[i]] <- lm(y ~ . , x_y)
}
```

We use multiple regression for every value in the Yi vector to be predicted. Hence if the Yi vector contains m values, we will build m svm models. The last line of code in the above snippet, trains a particular svm model by feeding the built data frame accordingly to the svm.

18.
```
series <- original_series
n=length(series)

X_now <- data.frame(x[1,])
colnames(X_now) <- columnNames[2:(m+1)]

predicted_time_series=vector(mode="list", length=n)
predicted_time_series[1]=X_now[1,1]
```

```
for(i in 2:n){
  X_next=X_now
  for(j in 1:m){
   X_next[1,j] = predict(models[[j]],newdata=X_now)
  }
  predicted_time_series[i]=X_next[1,1]
  X_now=X_next
}
```

In the beginning of this snippet the series will have only 70%(training dataset) of the time series values.The series is reinitialized to the original time series.

After training is done, for testing purpose, we have to predict the entire time series for from the first Xi vector. In this section the first vector X1 is initialized.

The predicted_time_series vector is initialized.

Use the X_now vector (initially it is X1), to predict the Xi+1 vector using the svm models created. The first value in Xi+1'th vector will be the i+1'th value in time series. Store that value in the predicted_time_series vector.

19.

```
retweet_count = cbind(series, predicted_time_series)
matplot(retweet_count,pch=c(1,2),col=c("green","red"), ylim=c(0,10000))
```

Here, 'series' contains the actual retweet time series values and 'predicted_time_series' contains the predicted time series values based on LS-SVM regression for the bottom 30 percent of the retweet time series file. THe graph is plotted using matplot where 'green' indicates original series and 'red' indicates the predicted time series.

20.

```
MAPE=0
for(i in 1:n){
  if(series[i] > 0){
   MAPE=MAPE+abs(predicted_time_series[[i]] - series[i])/series[i]
  }
}
MAPE=MAPE/n
return(MAPE)
```

Calculate MAPE value for a single retweet time series by taking the absolute value of difference between predicted and actual values divided by the length of the time series.

21.

```
listOfChaoticSeriesFiles <- list.files("chaoticSeriesFiles",
    pattern="*.csv", full.names=TRUE)
```

```
numberOfFiles <- length(listOfChaoticSeriesFiles)
```

'listOfChaoticSeriesFiles' is a list containing all chaotic time series files. 'numberOfFiles' contains the total number of such chaotic files.

22.

```
mape=0
faultCount=0
for(i in 1:numberOfFiles){
  t_mape = regressOn(listOfChaoticSeriesFiles[i])

  if(t_mape < 0 || t_mape > 10000){
    faultCount = faultCount + 1
  }
  else{
    mape = mape + t_mape
  }
}
print(paste0("Average Mape is : "mape/(numberOfFiles - faultCount)))
```

This block of code calculates the average mean absolute percentage error for all the retweet time series combined together. First, the regressOn function defined earlier returns a MAPE value for a particular retweet time series file. The fault count is incremented if the MAPE value is not applicable(less than 0 or greater than 10000). The current file MAPE value is added to the total MAPE value and later divided by the number of total chaotic time series files subtracted by the number of fault files, i.e., files which haven't been taken into consideration.

3 Results

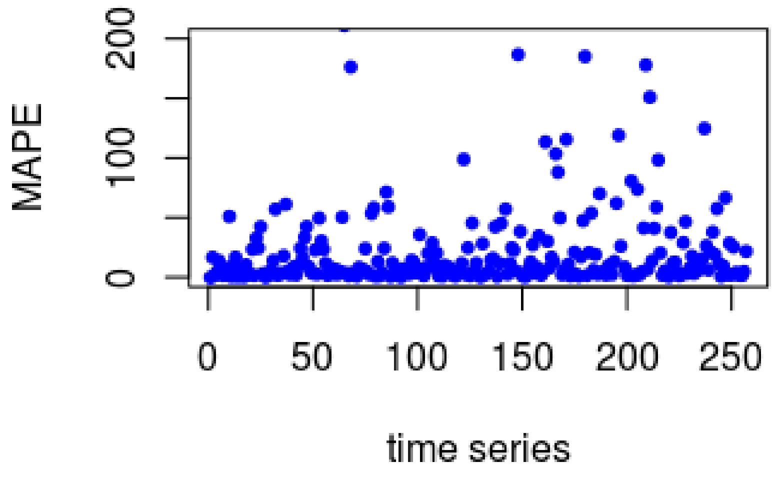

Figure 6: mape the 300 time series

The implementation was done on a Sina Weibo dataset, Sina Weibo is a twitter like website in china. We extracted time series of retweets covering over a period of 10 weeks.

The plot indicates the Mean Absolute Percentage Error(MAPE) of the 300 time series, obtained from a time delay of 60 minutes and a embedding dimension of 3. As it is evident from the graph, the MAPE of most of the time series lies on or below the value of 25.

Goodness of fit	
Model(embedding dimension)	MAPE
lssvm(2)	42.56
lssvm(3)	27.503
lssvm(4)	77.608
lssvm(5)	45.146
lssvm(6)	164.65
lssvm(7)	87.0

Table 1: Comparison of Goodness of Fit for different embedding dimensions

As it is clear from the table an embedding dimension of 3 gave the best results with an accuracy of 72.5%.

One other discovery during the project was that, for time series which are chaotic but have a very sudden spike in the number if retweets(1 to suddenly 4k), the proposed model works very badly. The MAPE for this type of series was 30000+. Which is a very large inefficiency. Hence our future work would be to analyse this behaviour in depth.

4 Conclusion and Future Work

This project deals with chaotic analysis of a time series. We use 0-1 chaos test to determine the chaotic behaviour of a retweet time series. If the series is found to be chaotic we apply a lssvm regression model on the series. We implemented the model on Sina Weibo dataset, and we got an accuracy of 72.5% in predicting the pattern of the time series. This system we have implemented can be used to predict and analyse the retweet or reposting pattern in social networking websites.

In the course of the project we found out that the proposed model works very poorly for retweet series with a very high sudden spike in the number of retweets, almost 1 to 4k. Our future work would be to analyse this behaviour and figure out a suitable model for the same problem.