Project Report On

Fuzzy GARCH Model for predicting Stock Market Volatility using GA

Submitted by

Poorva Rane(13IT126)

Balaji Balasubramanian(13IT207)

Nidhi Sridhar(13IT223)

VII SEM B.Tech (IT)

Under the guidance of

Prof. Biju R Mohan

in partial fulfillment for the award of the degree

Bachelor of Technology

In

Information Technology

At

Department of Information Technology

National Institute of Technology Karnataka, Surathkal

November 2016

1. Introduction

Often, while analyzing time-dependent data, conditional variances are not consistent with the assumption of homogeneity that is commonly associated with traditional econometrics models, especially those which treat financial data. Conditional variance plays an important role is in the phenomenon of volatility clustering. Volatility clustering means that large changes tend to follow other large changes, and small changes tend to follow small changes. Because of this the autoregression conditional heteroscedasticity (ARCH) model was introduced by Engle(1982). He adopted a model in which conditional variances in time-dependent data were subject to influences from previous unexpected factors. Furthermore, he assumed that the conditional variances were functions of error terms, allowing them to change over time. He proposed that the ARCH Model could solve the biases and therefore address traditional econometrics models.

Building on Engle's ARCH(q) model, Bollerslev (1986) made use of the Autoregressive Moving Average (ARMA) model to introduce the GARCH model. The GARCH model uses prior conditional variances to estimate the degree of transmission of volatility; it is characterized by a fat tail and excess kurtosis. Its ability to explain the transmission of volatility is a key advantage of this approach. For these reasons, the GARCH model is frequently used to explore the returns and transmissions of volatility in time-dependent financial data sets. However, financial assets are easily impacted by both positive and negative information, and the impacts are asymmetric. The GARCH model does not recognize transmissions of volatility that derive from the input of positive or negative information. Therefore, this model is not appropriate when the market is asymmetric.

To address this issue, we combine GARCH models and functional fuzzy systems. We apply these new models to real-world financial markets using GA. The process of optimizing functional fuzzy systems and GARCH model parameters is highly complex and nonlinear. A GA-based parameter estimation algorithm is proposed to derive the optimal solution for the fuzzy GARCH model.

2. Algorithm Involved

Genetic Algorithm (GA) is a method for optimizing machine learning algorithms inspired by the processes of natural selection and genetic evolution (Goldberg, 1989; Crefensteet, 1986; Holland, 1962). GA applies operators to a population of binary strings that encode the parameter space. A parallel global search technique emulates natural genetic operators such as reproduction, crossover, and mutation. At each generation, the algorithm explores different areas of the parameter space and then directs the search to the region where there is a high probability of finding improved performance. Because GA simultaneously evaluates many points in a parameter space, it is more likely to ultimately converge on the global solution. In particular, there is no requirement that the search space is differentiable or continuous, and the algorithm can iterate several times on each data point. Accordingly, it is a very suitable approach for time-varying nonlinear functions (Zhou & Khotanzad, 2007).

To use the genetic algorithm in the problem of fuzzy GARCH model parameter estimation, the relevant variables are first coded into a binary string called a chromosome. In each generation, three basic genetic operators (reproduction, crossover, and mutation) are performed to generate a new population with a constant population size. The chromosomes that remain after the population is reduced by the principle of survival of the fittest produce a better population candidate solution. The convergence of the proposed GA estimation scheme can be guaranteed via the theorem of the schema discussed in Holland (1975) and in Toroslu (2007). The estimation parameter that is obtained by the proposed estimation scheme ultimately converges to the optimal or near-optimal solution.

3. Problem Explanation

Consider a ARCH(q) model that is defined as (Engle, 1982)

$$
\begin{aligned}
y(t) &= a(t) \\
a(t) &= \sqrt{h(t)}\varepsilon(t) \\
h(t) &= \alpha_0 + \sum_{i=1}^{q} \alpha_i a^2(t-i)
\end{aligned}
$$

where *y(t)* is a random variable representing certain stock market data, *ε(t)* is a zero mean and unit variance white noise random process, *h(t)* is the conditional variance of *ε(t)*; t is the time index, and α_0, α_i are nonnegative; $\alpha_0 > 0$; $\alpha_0 \geq 0$.

Bollerslev (1986) modified the conditional variance term in the ARCH(q) model, by assuming that the conditional variances are influenced not only by the squared error terms, but also by previous conditional variances. He incorporated previous conditional variances into the process for estimating transmission of volatility. The result was his proposed GARCH(p, q) model. The model is defined as

$$
\begin{aligned}
y(t) &= a(t) \\
a(t) &= \sqrt{h(t)}\varepsilon(t) \\
h(t) &= \alpha_0 + \sum_{i=1}^{q} \alpha_i a^2(t-i) + \sum_{j=1}^{p} \beta_j h(t-j)
\end{aligned}
$$

where α_0, α_i and β_j are unknown parameters that must be estimated. Without loss of generality, we assume

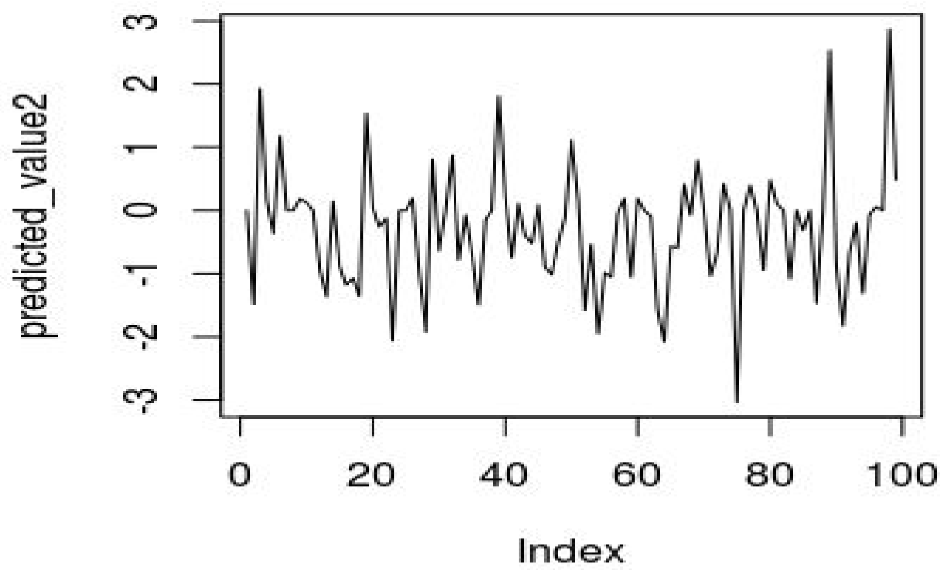
$$
\begin{aligned}
&\alpha_0 > 0, \alpha_i \geqslant 0; \quad i = 1, 2, \ldots, q;\ q > 0 \\
&\beta_j \geqslant 0; \quad j = 1, 2, \ldots, p;\ p > 0 \\
&\sum_{i=1}^{q} \alpha_i + \sum_{j=1}^{p} \beta_j < 1
\end{aligned}
$$

The conventional GARCH model has long been considered the most popular instrument for analyzing time-dependent data that features time-varying conditional variance. However, the model does not perform well when the market features clustering behavior. Ignoring this fact can lead to poor prognostic characteristics. Consequently, we adopt functional fuzzy systems to propose a new fuzzy GARCH model.

4. Design Procedure

The design procedure for the fuzzy GARCH model was divided into the following steps:

- Generate daily stock returns by taking the logarithmic difference of the input data and multiply by 100. y(t) = 100×log*S(t)* - log*S(t-1)*, where *S(t)* is the daily closing price at time t.
- Generate random population of P chromosomes.
- Decode each string into the corresponding parameter vector.
- Use the stability test, to $\sum_{i=1}^{q}\alpha_i + \sum_{j=1}^{p}\beta_j < 1$ specify the parameter domain of the coefficients of the GARCH model. If this fails, renew the chromosome.
- Calculate the fitness values according to the fitness equation, *Fit(θ) = 1/E(θ)*.
- Keep the best chromosomes intact for the next generation.
- Use reproduction, crossover, and mutation to generate new chromosomes in the next generation.
- We then obtain the predicted stock market values and plot them as shown in the graph below.



5. Code With Documentation

```

# Read dataset from .csv file
#DJA <- read.csv("C:\\Users\balaji\Desktop\DJA.csv")taset from .csv file
DJA <- read.csv("C:\\Users\\balaji\\Desktop\\DJA.csv")
# import library tseries
library("tseries", lib.loc="~/R/win-library/3.3")
library("GA", lib.loc="~/R/win-library/3.3")
#import GA library
#library("GA", lib.loc="~/R/x86_64-pc-linux-gnu-library/3.0")

#Store the daily closing values of NSADAQ dataset into stock_values
stock_values=DJA[1:2002,2]
# Generate daily stock-return series by taking the logarithmic difference of the daily stock index * 100
log_d=diff(log(stock_values))
log_d1=log_d*100
#Get the first 2000 values
in_data=log_d1[1:1999]


alpha0=c(0.1 ,0.2,0.3,0.4,0.5)
alpha1=c(0.1,0.2,0.3,0.4,0.5)
beta1=c(0.1,0.2,0.3,0.4,0.5)
center=array(data=(0.1) , dim=c(5,5))
spread=array(data=(0.1) , dim=c(5,5))
h=array(data=0,dim=c(1999))

# Get the heteroscedasticity for each fuzzy rule
get_h <-function(l){
  h[1]=alpha0[l]
  for(t in 2 : 1900){
    h[t] = alpha0[l]+(alpha1[l]*in_data[t-1]*in_data[t-1])+(beta1[l]*h[t-1])
  }
  return(h)
}

```

```

# Get membership values for each fuzzy rule
membership <- function(l){
  ul=array(0.1, dim=c(1900))
  for(t in 6:1900){
    init = exp((-0.5)*((in_data[t-1]-center[l,1])/spread[l,1])^2)
    for(j in 2:5){
      ul[t] = init * exp((-0.5)*((in_data[t-1]-center[l,j])/spread[l,j])^2)
      init = ul[t]
    }
  }
  return(ul)
}


# Set the fitness function
parameters=array(0.1,dim=c(65))

# Create arrays for storing intermediate values
gl = array(0.1,dim=c(5,1999))
sum_ht = array(0.1,dim=c(5,1999))

predicted_value=array(0.1,dim=c(1900))

fitness <- function(parameters)
{
  center=array(data=(0.1) , dim=c(5,5))
  spread=array(data=(0.1) , dim=c(5,5))

  center[1,]=parameters[1:5]
  center[2,]=parameters[6:10]
  center[3,]=parameters[11:15]
  center[4,]=parameters[16:20]
  center[5,]=parameters[21:25]

  spread[1,]=parameters[26:30]
```

```r
  spread[3,]=parameters[36:40]
  spread[4,]=parameters[41:45]
  spread[5,]=parameters[46:50]

  alpha0=parameters[51:55]
  alpha1=parameters[56:60]
  beta1=parameters[61:65]
  #h=array(data=0,dim=c(1999))

  # Summation of membership values
  sum_membership = membership(1)+membership(2)+membership(3)+membership(4)+membership(5)



  # Calculate summation of h
  for(j in 1:5){
    temp_membership = membership(j)
    temp_h=get_h(j)
    #   print(temp_h)
    for(i in 1:1900){
      gl[j,i]=temp_membership[i]/sum_membership[i]
      sum_ht[j,i]=gl[j,i]*temp_h[i]
      #    print(temp_h[i])
    }
  }

  #Create white noise process
  noise <- rnorm(1900)
  scale <- function(noise){(noise-min(noise))/(max(noise)-min(noise))*(max(in_data)-min(in_data))+min(in_data)}
  w.noise <- scale(noise)

  # Predict stock market data using Fuzzy GARCH

  for(i in 1:1900){
    predicted_value[i] = sqrt(sum(sum_ht[,i]))*w.noise[i]
    if(is.nan(predicted_value[i])){
```


```r
  for(i in 1:1900){
    predicted_value[i] = sqrt(sum(sum_ht[,i]))*w.noise[i]
    if(is.nan(predicted_value[i])){
```


```r
      predicted_value[i] = 0
    }
  }

  #print(predicted_value)

  return(sum((in_data[1:1900]-predicted_value)^2))

}

# Set min and max values for GA
c0=c(-0.07,-0.07,-0.07,-0.07,-0.07)
c1=c2=c3=c4=c0
s0=c(0,0,0,0,0)
s1=s2=s3=s4=s0
alpha0=c(0,0,0,0,0)
alpha1=beta1=alpha0
min_f=c(c0,c1,c2,c3,c4,s0,s1,s2,s3,s4,alpha0,alpha1,beta1)

c0=c(0.07,0.07,0.07,0.07,0.07)
c1=c2=c3=c4=c0
s0=c0
s1=s2=s3=s4=s0
alpha0=c(0.5,0.5,0.5,0.5,0.5)
alpha1=alpha0
beta1=c(1,1,1,1,1)
max_f=c(c0,c1,c2,c3,c4,s0,s1,s2,s3,s4,alpha0,alpha1,beta1)

# Call GA with the defined fitness function
GA <- ga(type = "real-valued",
```

```r
# Call GA with the defined fitness function
GA <- ga(type = "real-valued",
         fitness = function(x) 1/fitness(parameters),
         min = min_f, max = max_f,
         popSize = 30, maxiter =100,
         pcrossover = 0.8,
         pmutation = 0.1)

# solution stores our parameter values
parameters=summary(GA)$solution


center=array(data=(0.1) , dim=c(5,5))
spread=array(data=(0.1) , dim=c(5,5))
center[1,]=parameters[1:5]
center[2,]=parameters[6:10]
center[3,]=parameters[11:15]
center[4,]=parameters[16:20]
center[5,]=parameters[21:25]

spread[1,]=parameters[26:30]
spread[2,]=parameters[31:35]
spread[3,]=parameters[36:40]
spread[4,]=parameters[41:45]
spread[5,]=parameters[46:50]

alpha0=parameters[51:55]
alpha1=parameters[56:60]
beta1=parameters[61:65]

#h=array(data=0,dim=c(1990))

membership2 <- function(l){
  ul=array(0.1, dim=c(99))
  for(t in 1901:1999){
    init = exp((-0.5)*((in_data[t-1]-center[l,1])/spread[l,1])^2)
```

```r
# Summation of membership values
sum_membership2=array(0.1, dim=c(99))
sum_membership2 = membership2(1)+membership2(2)+membership2(3)+membership2(4)+membership2(5)

get_h2 <-function(l){
  #h2=array(0, dim=c(99))
  #print(h[1900])
  t_temp=get_h(l)
  for(t in 1901 : 1999){
    t_temp[t] = alpha0[l]+(alpha1[l]*in_data[t-1]*in_data[t-1])+(beta1[l]*t_temp[t-1])
    # print(h[t])
  }
  return(t_temp)
}

# Calculate summation of h
for(j in 1:5){
  temp_membership = membership2(j)
  #print(temp_membership)
  temp_h=get_h2(j)
  #print(temp_h)
  for(i in 1901:1999){
    # print(sum_membership2[i-1900])
    gl[j,i]=temp_membership[i-1900]/sum_membership2[i-1900]
    #print (gl[j,i])
    sum_ht[j,i]=gl[j,i]*temp_h[i]
    # print(sum_ht[j,i])
  }
}

#Create white noise process
noise <- rnorm(1999)
scale <- function(noise){(noise-min(noise))/(max(noise)-min(noise))*(max(in_data)-min(in_data))+min(in_data)}
w.noise <- scale(noise)

# Predict stock market data using Fuzzy GARCH
```


```r
w.noise <- scale(noise)

# Predict stock market data using Fuzzy GARCH
```


```r
predicted_value2=array(0.1,dim=c(99))
for(i in 1:99){
  predicted_value2[i] = sqrt(sum(sum_ht[,1900+i]))*w.noise[1900+i]
  #print(predicted_value2[i])
  if(is.nan(predicted_value2[i])){
    predicted_value2[i] = 0
  }
}

print(sum((in_data[1901:1999]-predicted_value2)^2))
plot(in_data[1901:1999],type='l')
plot(predicted_value2,type='l')
```