Advanced Time Series Analysis (IT 833)

Submitted by: Sampat Kr Ghosh
Roll Number: 192IT020
Submitted to: Dr. Biju R Mohan

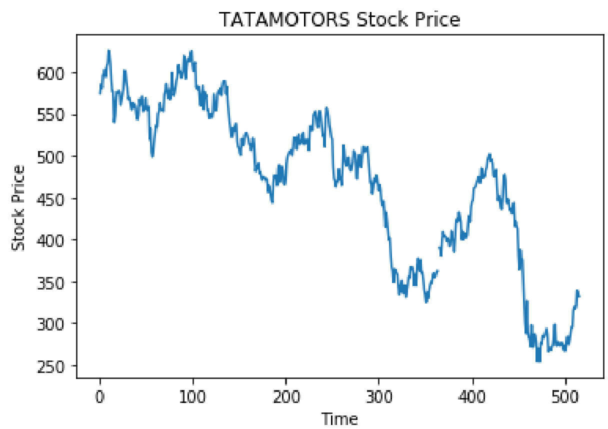
Assignment Report for Question 1

Introduction

A framework called the log-periodic power law (LPPL) model has gained a lot of attention with the many successful predictions it made Johansen et al. [1] proposed the LPPL model, which assumes that there exist two types of agents in the market: a group of traders with rational expectations and a group of noise traders with herding behaviour. The noise traders are organised into networks, and they tend to imitate others. At the macro level, all the agents will continue investing where arbitrage is limited because rational traders lack the knowledge about the time of crash and are assumed to be risk-neutral. It is still rational for them to invest on speculative assets because the risk of a crash is compensated for by the profits. As a consequence, rational traders will self-limit their arbitrage behaviour. The herding behaviour of noise traders is the origin of the positive feedback process; that is, given a high price, the imitation among noise traders leads to increased demand, which pushes the price further up. In this project, we have optimized the parameters of LPPL to predict stock price crash. **We are following the algorithms given in the paper titled "Forecasting Financial Crashes: Revisit to Log-Periodic Power Law" by Bingcun Dai, Fan Zhang, Domenico Tarzia and Kwangwon Ahn.**

Datasets

Stock Price dataset was taken from here. TATAMOTORS dataset from May 2, 2018 to June 9, 2020.



LPPL Overview

The LPPL model is an oscillating, exponential model for price evolution. The intuition that underlies crash prediction is essentially the idea of the impossibility for continuing exponential price growth, with increasing oscillations approaching failure indicated by swings in investor sentiment. Mathematically, a basic LPPL model proposes that the price of an asset evolves at time **t** according to:

$$\ln[p(t)] \approx A + B_0(t_c - t)^{\beta}\{1 + C\cos[\omega \ln(t_c - t) + \phi]\}$$

where:

ln[P(t)] – natural log of Price **p** at time **t**

$\mathbf{t_c}$ – the "Critical Time" period i.e. the time of most probable crash

$\boldsymbol{\beta}$ (beta) – the exponential price growth with constraint $\mathbf{0 < \beta < 1}$

$\boldsymbol{\omega}$ (omega) – the oscillation amplitude with general constraint $\mathbf{2 < \omega < 20}$

ϕ (phi) – a fixed phase constant parameter with constraint $\mathbf{0 < \phi < 2\pi}$

A – a constant equal to the log price at the Critical time (**tc**) (ie. **ln[P(tc)] > 0**)

B(o) – a constant embodying the scale of power law where **B(o) <0**

C – a constant that captures the magnitude of the oscillation around price growth where**|C| < 1**

The conditions for $\boldsymbol{\beta}$ and $\boldsymbol{\omega}$ indicate "faster-than--exponential" acceleration of the log-price. The condition for ϕ expresses that the oscillations are neither too slow (such that they would simply become part of the trend) or too fast (such that they would fit the random component).

The above equation has *7 parameters* and 3 of the key ones ($\boldsymbol{\beta}$, $\boldsymbol{\omega}$ and $\boldsymbol{\phi}$) are non-linear. Therefore estimating LPPL models in general has never been easy due to the sheer number of parameters involved. However, rewriting and simplifying the LPPL equation as:

$$y_i = A + Bf_i + Cg_i$$

where

$$y_i = \ln I_i \quad \text{or} \quad I_i \ , \qquad f_i = (t_c - t_i)^\beta$$

$$g_i = (t_c - t_i)^\beta \cos(\omega \ln(t_c - t_i) + \phi)$$

then we see that the linear parameters A, B, and C can be obtained by using ordinary least squares regression. Reducing the number of parameters from 7 to 4 helps simplify the calibration problem of the model.

The optimal values of the final 4 parameters (namely β, ω, ϕ and T_c) are found using a metaheuristic search algorithm, Grey Wolf Optimizer. However, a Simulated Annealing algorithmic approach was used here due to ease of implementation but the goal is the same, ie. using a probabilistic technique to approximate global optimization in a large search space. However, being metaheuristic, there is no guarantee of finding an optimum or near optimum solution of the cost function.

Differential Evolution Optimizer

Differential evolution (DE) is a type of evolutionary algorithm developed by Rainer Storn and Kenneth Price for optimization problems over a continuous domain. The prime idea of DE is to adapt the search during the evolutionary process. During the initial stage of evolution, the perturbations are large since parent individuals are far away from each other. As the evolutionary process matures, the population converges to a small region, and the perturbations adaptively become small. Hence, the DE performs a global exploratory search during the early stages of the evolutionary process and local exploitation during the mature stage of the search.

Methodology

The algorithm used to detect crash and optimized the parameters are as follows:

1. Detect the peaks of the sample with window size κ.
2. Assign the distance-based weight to each peak.
3. Randomly select three consecutive peaks based on the weights.
4. Use these three consecutive peaks for price gyration and obtain the initial values for t_c, ω, and ϕ from the price gyration as $t_c = \rho k - j / \rho - 1$, $\omega = 2\pi/\ln \rho$, and $\phi = \pi - \omega \ln t c - k$ with $\rho = j - i / k - j$.
5. Set the initial values $\beta = 1$ and C = 0, and estimate the initial values of A and B using Linear Regression.

$$y_t = A + B(t_c - t) + \varepsilon_t$$

6. Repeat steps 3 to 5, and obtain a series of initial values for the seven LPPL parameters.
7. Find the LPPL parameters using DE, with the initial population of the parameters from step 6 by minimizing the objective function given below:

$$\text{RMSE} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(y_t - Y_t)^2}$$

where y_t and T denote the log of observation at time t and the number of trading days in the dataset and Y_t is the predicted log of price at time t.

8. Repeat steps 1 to 7 with changing the window size κ, and obtain the prediction interval for the critical time t_c

At the end the t_c with the lowest RMSE will be the point of crashing.

Parameters Used

Population Size for DE was kept at 70. Maximum number of iterations was 100. And the starting window size was 10.

Results

The crashing point for the given data set is 366.0487804878049 days with RMSE of .08087070010686072.

Tools used

1. Python 3
2. Pandas package
3. Numpy package
4. Scikit Learn package
5. Matplotlib package