# Untitled

Sabato Gargiulo

17/11/2020

## LPPL EQUATION AND FORMULATION

The fundamental equation of LPPL model for the expected logarithm of price is:

$$E\big[\ ln(P_t)\big] = A + B(t_c - t)^m + C(t_c - t)^m \cos(\omega\ ln(t_c - t) - \phi) \quad con \quad A = ln(P_{t_c})$$

Il metodo di stima proposto da Sornette parte dall'espansione del termine coseno come segue:

$$C\cos(\omega\ ln(t_c - t) - \phi) = C\cos(\omega\ ln(t_c - t))\cos(\phi) + C\sin(\omega\ ln(t_c - t))\sin(\phi)$$

Se indichiamo con $C_1 = C\cos(\phi)$ e $C_2 = C\sin(\phi)$ possiamo ridurre i parametri non lineari da $\{m, \omega, t_c, \phi\}$ a $\{m, \omega, t_c\}$ mentre i parametri lineari passano da $\{A, B, C\}$ a $\{A, B, C_1, C_2\}$. Usando la nuova formulazione il logaritmo del prezzo può essere modellato come:

$$LPPL(\phi, t) = A + B(f) + C_1(g) + C_2(h)$$

ove $\phi = \{A, B, C_1, C_2, m, \omega, t_c\}$ è il vettore di parametri che ci interessa stimare e :

$$f = (t_c - t)^m$$
$$g = (t_c - t)^m \cos(\omega\ ln(t_c - t))$$
$$h = (t_c - t)^m \sin(\omega\ ln(t_c - t))$$

Per stimare i parametri utilizziamo il metodo least-squares con funzione di costo:

$$F(t_c, m, \omega, A, B, C_1, C_2) = \sum_{i=1}^{N} \Big[ ln(p_{\tau_i}) - A - B(t_c - t)^m - C_1(t_c - \tau_i)^m \cos(\omega\ ln(t_c - \tau_i)) - C_2(t_c - \tau_i)^m \sin(\omega\ ln(t_c - \tau_i)) \Big]^2$$

Vincolando i 4 parametri lineari ai 3 parametri non lineari otteniamo il seguente problema di ottimo

$$\{\hat{t_c}\ , \hat{m},\ \hat{\omega}\} = \arg \min_{t_c, m, \omega}\ F_1(t_c, m, \omega)$$

dove

$$F_1(t_c, m, \omega) = \min_{A, B, C_1, C_2}\ F(t_c, m, \omega, A, B, C_1, C_2)$$

Infine il problema di ottimo

$$\{\hat{A}, \hat{B}, \hat{C_1}, \hat{C_2}\} = \arg \min_{A,B,C_1,C_2} F(t_c, m, \omega, A, B, C_1, C_2)$$

la cui soluzione si ottiene dalla seguente equazione matriciale:

$$\begin{pmatrix} N & \sum f_i & \sum g_i & \sum h_i \\ \sum f_i & \sum f_i^2 & \sum f_i g_i & \sum f_i h_i \\ \sum g_i & \sum f_i g_i & \sum g_i^2 & \sum g_i h_i \\ \sum h_i & \sum f_i h_i & \sum g_i h_i & \sum h_i^2 \end{pmatrix} \begin{pmatrix} \hat{A} \\ \hat{B} \\ \hat{C_1} \\ \hat{C_2} \end{pmatrix} = \begin{pmatrix} \sum y_i \\ \sum y_i f_i \\ \sum y_i g_i \\ \sum y_i h_i \end{pmatrix}$$

con

$$y_i = ln(p_i)$$
$$f_i = (t_c - t_i)^m$$
$$g_i = (t_c - t_i)^m \cos(\omega\ ln(t_c - t_i))$$
$$h_i = (t_c - t_i)^m \sin(\omega\ ln(t_c - t_i))$$

```
library(cmaes)
library(quantmod)
library(zoo)
library(alphavantager)
library(lubridate)
library(dplyr)
library(parallel)
library(doParallel)

#setwd("C:\\Users\\gargi\\Desktop\\Tesi SSF\\Code")
setwd("~/Desktop/Tesi SSF/Code")

filename <- "SP500.csv"
filepath <- paste("./data/", filename, sep="")
filesname <- substr(filepath, nchar("./data/")+1, nchar(filepath)-4)



ticker <- read.csv(filepath)
ticker$Date <- as.Date(ticker$Date, format = "%Y-%m-%d")
plot(ticker$Date,ticker$Adj.Close ,type="l")

ticker <- ticker[,c(1,6)]


ticker$t <- decimal_date(ticker$Date)
names(ticker) <- c("Date", "Close", "t")



date_txt_to_base = "2007-9-28"
to_base <- as.Date(date_txt_to_base)

date_txt_from <- as.character(to_base-860)
from_base <- as.Date(date_txt_from)
```

```r
#RIMUOVIAMO RIGHE CON NA

ticker$Close <- na_if(ticker$Close,"null")
ticker <- na.omit(ticker)
ticker$Close <- as.numeric(ticker$Close)

#Slaving Linear Variables
LPPL <- function(data, m=1, omega=1, tc=0) {
  data$X <- tc - data$t
  data$Xm <- data$X ** m #B
  data$Xm.cos <- data$X ** m * cos(omega * log(data$X)) #C1
  data$Xm.sin <- data$X ** m * sin(omega * log(data$X)) #C2
  data$logP <- log(data$Close)
  return(lm(logP ~ Xm + Xm.cos + Xm.sin, data=data))
}

#Initial Estimates of A, B, C1 and C2 through Least Squares
FittedLPPL <- function(data, lm.result, m=1, omega=1, tc=0) {
  data$X <- tc - data$t
  A <- lm.result$coefficients[1]
  B <- lm.result$coefficients[2]
  C1 <- lm.result$coefficients[3]
  C2 <- lm.result$coefficients[4]
  result <- exp(A + B * (data$X ** m) +
                  C1 * (data$X ** m) * cos(omega * log(data$X)) +
                  C2 * (data$X ** m) * sin(omega * log(data$X)))
  return(result)
}


#Rewritten for plotting
FittedLPPLwithexpected <- function(data, lm.result,
                                   x_vector, m=1, omega=1, tc=0) {
  tmp_vector <- tc - x_vector
  A <- lm.result$coefficients[1]
  B <- lm.result$coefficients[2]
  C1 <- lm.result$coefficients[3]
  C2 <- lm.result$coefficients[4]
  result <- exp(A + B * (tmp_vector ** m) +
                  C1 * (tmp_vector ** m) * cos(omega * log(tmp_vector)) +
                  C2 * (tmp_vector ** m) * sin(omega * log(tmp_vector)))
  return(result)

}

#Function for getting final values of A, B, C1 and C2 parameters
getlinear_param <- function(m, omega, tc) {
```

```
  lm.result <- LPPL(rTicker, m, omega, tc)
  getcoeff_regLPPL <- c(lm.result$coefficients[1],lm.result$coefficients[2],
                        lm.result$coefficients[3], lm.result$coefficients[4])
}


tryParams <- function (m, omega, tc) {
  lm.result <- LPPL(rTicker, m, omega, tc)
  plot(rTicker$t, rTicker$Close, typ='l') #Plot graph
  generate_vector = seq(min(rTicker$t), tc-0.002, 0.002)
  lines(generate_vector, FittedLPPLwithexpected(rTicker,
                                                lm.result,
                                                generate_vector,
                                                m,
                                                omega, tc), col="red")
}


residuals_with_ts <- function(ts, m, omega, tc) {
  lm.result <- LPPL(ts, m, omega, tc)
  return(sum((FittedLPPL(ts, lm.result, m, omega, tc) - ts$Close) ** 2))
}


residuals_with_ts_obj <- function(x, ts) {
  return(residuals_with_ts(ts, x[1], x[2], x[3]))
}

#nbre_step_backward <- 720
#nbre_generation <- 80
nbre_step_backward <- 720
nbre_generation <- 80

fitter_tt <- function(i, from_base, ticker,
                      date_txt_from, to_base
                      ){




  from <- from_base+i
  vec_control <- data.frame(maxit = c(nbre_generation))

  # if (as.POSIXlt(from)$wday != 0 & as.POSIXlt(from)$wday != 6) {
  .GlobalEnv$ticker <- ticker
  rTicker <- base::subset(ticker, ticker$Date >= from & ticker$Date <= to_base)

  last_row <- tail(rTicker, 1)
  first_row <- head(rTicker, 1)
  dt <- last_row$t -first_row$t

  test <- cmaes::cma_es(c(0.01, 5, max(rTicker$t)+0.002), residuals_with_ts_obj,
                        rTicker, lower=c(0.1, 1, max(rTicker$t)+0.002),
```

```
                        upper=c(2, 50, max(rTicker$t)+0.2*dt),
                        control=vec_control)

  linear_param <- getlinear_param(test$par[1], test$par[2], test$par[3])


  fitted <- FittedLPPLwithexpected(rTicker,
                                   LPPL(rTicker,
                                        test$par[1],
                                        test$par[2],
                                        test$par[3]),
                                   last_row$t, test$par[1],
                                   test$par[2], test$par[3])


  df_result <- c(date_txt_from, format(to_base, "%Y-%m-%d"), last_row$t, first_row$t,
                 last_row$Close,
                 fitted,
                 -i,
                 nbre_generation,
                 test$par[3]-last_row$t,
                 as.integer((test$par[3]-last_row$t)/(1/365)),
                 test$par[1],
                 test$par[2],
                 test$par[3],
                 linear_param[1],
                 linear_param[2],
                 linear_param[3],
                 linear_param[4],
                 (test$par[2]/2)*log(abs((test$par[3]-first_row$t)/(dt))),
                 (test$par[1]*abs(linear_param[2]))/(test$par[2]*
                                                       abs(atan(linear_param[4]/
                                                                  linear_param[3])
                                                           )),
                 (last_row$Close-fitted)/fitted )
  #tryParams(test$par[1], test$par[2], test$par[3])
  return(df_result)

}


cl <- parallel::makeForkCluster(6)
doParallel::registerDoParallel(cl)



#Loop for weekly collapsing windows
df_result <- foreach (i = seq(0,nbre_step_backward,5), .combine = rbind) %dopar%
  {

  .GlobalEnv$ticker <- ticker
  fitter_tt(i, from_base,ticker,date_txt_from,to_base)
```

```
}
#}

parallel::stopCluster(cl)

df_result <- as.data.frame(df_result)
i <- 3:20
df_result[3:20] <- lapply(df_result[3:20],as.numeric)

ticker[4:7] <- vector("numeric",length=nrow(ticker))

colnames(df_result) <- c("date_from", "date_to", "t2","t1", "price",
                         "fitted price", "step_backward", "nbre_generation",
                         "t_until_critical_point", "days_before_critical_time",
                         "m","omega", "tc", "A", "B", "C1", "C2", "oscill",
                         "damp","rel_err")

colnames(ticker)[4:7] <- c("early_warn_lt","bubble_end_lt",
                           "early_warn_st","bubble_end_st")


#Condizione early warning (LONG TIME) "Ear_lt"
ticker[which(ticker$Date == to_base),4] <- nrow(as_tibble(df_result)[1:125,] %>%

               filter(m >= 0.01 & m <= 1.2 & omega >=2 & omega <= 25
                      & tc <= t2+0.1*(t2-t1) & oscill >= 2.5 & damp >=0.8
                      & rel_err >=0 & rel_err <=0.05))/125
#Condizione end flag (LONG TIME) "End_lt"

ticker[which(ticker$Date == to_base),5] <- nrow(as_tibble(df_result)[1:125,] %>%

                  filter(m >= 0.01 & m <= 0.99 & omega >=2 & omega <= 25
                         & tc <= t2+0.1*(t2-t1) & oscill >= 2.5 & damp >=1
                         & rel_err >=0 & rel_err <=0.2))/125

# Condizione early warning (SHORT TIME) "Ear_st"
ticker[which(ticker$Date == to_base),6] <- nrow(as_tibble(df_result)[126:145,] %>%

                   filter(m >= 0.01 & m <= 1.2 & omega >=2 & omega <= 25
                          & tc <= t2+0.1*(t2-t1) & oscill >= 2.5 & damp >=0.8
                          & rel_err >=0 & rel_err <=0.05))
#Condizione end flag (SHORT TIME) "End_st"

ticker[which(ticker$Date == to_base),7] <- nrow(as_tibble(df_result)[126:145,] %>%

                 filter(m >= 0.01 & m <= 0.99 & omega >=2 & omega <= 25
                        & tc <= t2+0.1*(t2-t1) & oscill >= 2.5 & damp >=1
                        & rel_err >=0 & rel_err <=0.2))/20
```

```
nowdatetime <- paste(format(Sys.Date(), "%Y%m%d"),
                     format(Sys.time(), "%H%M%S"),
                     sep="_")

write.csv(df_result,
          paste('./data/', filesname, '_analysis_done_on_', nowdatetime,
                "_from_", date_txt_from, "_to_", date_txt_to_base, ".csv",
                sep=''))


rm(list())
```

# DLPPL Confidence

Per quanto riguarda la grandezza della finestra, distinguiamo in indicatore LPPL di lungo termine e breve termine. Fissato il tempo "presente" $t_2$, stimiamo il modello sulla finestra temporale $(t_1, t_2)$, di lunghezza $dt = t_2 - t_1$. Il modello sarà stimato su finestre temporali che diminuiscono il lunghezza da 750 giorni a 125, con step di 5 giorni per quanto riguarda la stima di lungo termine. Per quella di breve termine il procedimento è lo stesso, con la differenza che $dt$ andrà da 125 a 20 giorni.

| **Item** | **Notation** | **Search space** | **Filtering condition (Early Warning)** | **Filtering Condition (End Flag)** |
|---|---|---|---|---|
| Nonlinear Parameters | $m$<br>$\omega$<br>$t_c$ | $[0, 2]$<br>$[1, 50]$<br>$[t_2 - 0.2dt, t_2 + 0.2dt]$ | $[0.01, 1.2]$<br>$[2, 25]$<br>$[t_2 - 0.05dt, t_2 + 0.1dt]$ | $[0.01, 0.99]$<br>$[2, 25]$<br>$[t_2 - 0.05dt, t_2 + 0.1dt]$ |
| Number of oscillations | $\frac{\omega}{2} \ln \left\lvert \frac{t_c - t_1}{t_2 - t_1} \right\rvert$ | — | $[2.5, +\infty]$ | $[2.5, +\infty]$ |
| Damping | $\frac{m\lvert B \rvert}{\omega \lvert C \rvert}$ | — | $[0.8, +\infty]$ | $[1, +\infty]$ |
| Relative Error | $\frac{p_t - \hat{p}_t}{\hat{p}_t}$ | — | $[0, 0.05]$ | $[0, 0.2]$ |

**Table 2.1:** Search space and filtering conditions to quantify valid LPPLS fits

Figure 1: Filtering condition

Per ogni finestra verranno quindi calcolati gli indicatori di cui sopra. Se tale finestra rientra nelle condizioni di filtering verrà considerata, altrimenti no.

Una volta filtrato ci basterà calcolare la frazione di finestre valide su quelle totali e moltiplicare questo valore per il segno della mediana dei rendimenti cumulati sulla finestra considerata per distinguere tra bolle positive e negative.

L'early bubble warning è un indicatore che serve a indentificare una bolla nella fase iniziale della sua formazione. Viceversa il "bubble end flag" ci indica quando una bolla è vicina alla propria fine poichè avverrà

un cambio di regime. Se il valore assoluto di questo indicatore è elevato, significa che il regime in corso non è sostenibile.

!! Nota Per ricavare C attraverso C1 e C2 faccio così:

$$C_1 = C\cos(\phi) \quad C_2 = C\sin(\phi) \quad ricordando \;\; che \;\; \sin(\phi)^2 + \cos(\phi)^2 = 1$$

Possiamo scrivere:

$$C_1^2 + C_2^2 = C^2(\sin(\phi)^2 + \cos(\phi)^2) \;\; quindi \;\; C = \sqrt{C_1^2 + C_2^2}$$

Potevo anche procedere ricavando $\phi$ :

$$\frac{C_1}{\cos(\phi)} = \frac{C_2}{sin(\phi)} \qquad \tan(\phi) = \frac{\sin(\phi)}{\cos(\phi)} = \frac{C_2}{C_1} \qquad \phi = \arctan\left(\frac{C_2}{C_1}\right)$$

E sostituendo $\phi$:

$$C = \frac{C_1}{\cos(\arctan(C_2/C_1))}$$