Markov Switching

[1]:
```
from IPython.display import display, Math, Latex
```

1 Descripción

El archivo MS_FAE, posee el programa MarkovSwitching_FAE.py, el cual es una versión hecha en base a los archivos del package statsmodels, en particular de 'regime_switching'. Este programa es una versión propia, que soporta los resultados y Wrap desde markov_switching.py. Además este archivo posee un modulo auxiliar, llamado Funciones.py, el cual tiene algunas funciones de utilidad para el programa en si. Es por esto que deben mantenerse dentro del mismo archivo.

Obs: 1. El programa fue escrito con interprete de python 3.6.9 64-bit
2. El sistema operativo utilizado fue Ubuntu 18.04 LTS, aunque esto no debe ser una preocupación mayor y solo es mencionado como documentación.
3. Para instalar y administrar un package escrito en Python, recurrimos a 'pip'. Para esto basta solo escribir *'pip install nombre_package'*
4. Para actualizar un package, escribimos *'pip install nombre_package --upgrade'*
5. En general las librerias mas utilizadas fueron **numpy** versión 1.18.2, **pandas** versión 1.0.3 y **statsmodels** versión 0.11.1.
6. Toda la documentación se puede encontrar en https://www.stata.com/manuals14/tsmswitch.pdf

[2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from MarkovSwitching_FAE import MarkovSwitching_FAE
```

[ ]:
```
from pandas_datareader.data import DataReader
from datetime import datetime

usrec = DataReader('USREC', 'fred', start=datetime(1947, 1, 1),␣
 ↪end=datetime(2013, 4, 1))
```

2 Hamilton's MarkovSwitching Model of Business Fluctuations

En los modelos de Markov-Switching de fluctuaciones comerciales, el punto de inflexión se trata como un evento estructural inherente al proceso de generación de datos. Una característica importante de tales modelos es que pueden capturar una forma particular de dinámica no lineal o de asimetría en las fluctuaciones comerciales. Por ejemplo, Hamilton (1989) permite que la media del crecimiento en el GNP real evolucione de acuerdo con un proceso de Markov-Switching de dos estados, permitiendo así que la dinámica de las recesiones sea cualitativamente distinta de la de las expansiones. El crecimiento en el PNB real se modela como un proceso $AR(4)$:

$$y_t - \mu_{S_t} = \phi_1(y_{t-1} - \mu_{S_{t-1}}) + \phi_2(y_{t-2} - \mu_{S_{t-2}}) + \phi_3(y_{t-3} - \mu_{S_{t-3}}) + \phi_4(y_{t-4} - \mu_{S_{t-4}}) + \epsilon_t$$

Donde $\epsilon_t \sim$ i.i.d$N(0, \sigma^2)$ y $\mu_{S_t} = \mu_0(1 - S_t) + \mu_1 S_1$. Además, las raices de $\phi(L) = (1 - \phi_1 L - \ldots - \phi_4 L^4) = 0$ viven fuera del circulo unitario, donde L is el 'lag operator' $Ly_t = y_{t-1}$ y y_t es el 'log' del GNP.

En cada periodo, el 'regimen transitions' de acuerdo a la siguiente matriz de transición,

$$\mathbb{P}(S_t = s_t \mid S_{t-1} = s_{t-1}) = \begin{bmatrix} p_{00} & p_{10} \\ p_{01} & p_{11} \end{bmatrix}$$

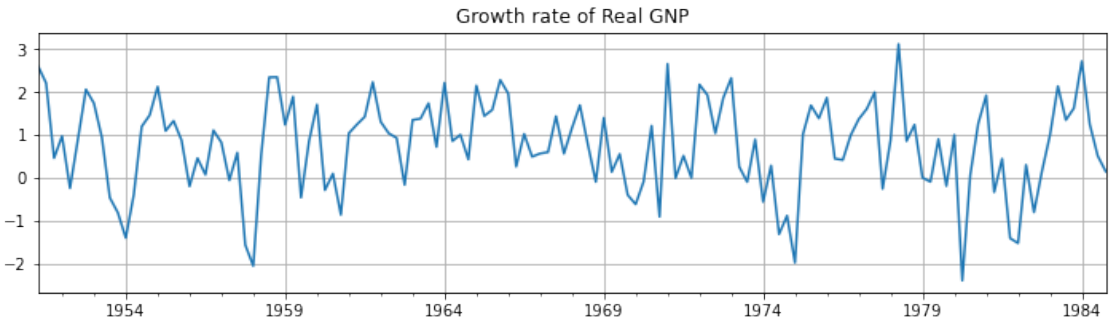
Donde p_{ij} es la probabilidad de transición del regimen i al regimen j.

Para replicar los resultados presentados en *State-Space Models, Kim & Nelson*, para el real GNP, primero obtenemos la data y generamos una serie y su respectivo plot.

[4]:
```
from statsmodels.tsa.regime_switching.tests.test_markov_autoregression import␣
 ↪rgnp
# Data RGNP, Hamilton
dta_hamilton = pd.Series(rgnp, index=pd.date_range('1951-04-01', '1984-10-01',␣
 ↪freq='QS'))
dta_hamilton.head()
```

[4]:
```
1951-04-01    2.593164
1951-07-01    2.202171
1951-10-01    0.458276
1952-01-01    0.968744
1952-04-01   -0.241308
Freq: QS-JAN, dtype: float64
```

[5]:
```
# Plot the data
dta_hamilton.plot(title='Growth rate of Real GNP', figsize=(12,3))
plt.grid()
```



Acá debemos detenernos y separar el proceso. Primero, creamos el objeto MarkovSwitching_FAE, con sus respectivos parametros, ya sea numero de observaciones *nobs*, la cantidad de regimenes *k_regimes*, etc. El punto importante es que k_params, nos entrega el numero de parametros que utilizaremos.

[6]:
```
mod_hamilton = MarkovSwitching_FAE(dta_hamilton, k_regimes=2, order=4,␣
↪switching_ar=False)
```

Una vez que el objeto se encuentra creado, se procede a realizar el ajuste con la función fit(). Uno de los puntos principales es la obtención de los parametros,

[7]:
```
def start_params(mod_hamilton):
    """
    (array) Parámetros iniciales para la estimación de máxima verosimilitud.
    """
    params = np.zeros(self.k_params, dtype=np.float64)

    # Transition probabilities
    if self.tvtp:
        params[self.parameters['regime_transition']] = 0.
    else:
        params[self.parameters['regime_transition']] = 1. / self.k_regimes

    endog = self.endog.copy()
    if self._k_exog > 0 and self.order > 0:
        exog = np.c_[self.exog, self.exog_ar]
    elif self._k_exog > 0:
        exog = self.exog
    elif self.order > 0:
        exog = self.exog_ar
    if self._k_exog > 0 or self.order > 0:
        beta = np.dot(np.linalg.pinv(exog), endog)
        variance = np.var(endog - np.dot(exog, beta))
    else:
        variance = np.var(endog)
```

```
    # Regression coefficients
    if self._k_exog > 0:
        if np.any(self.switching_coeffs):
            for i in range(self.k_regimes):
                params[self.parameters[i, 'exog']] = (
                        beta[:self._k_exog] * (i / self.k_regimes))
        else:
            params[self.parameters['exog']] = beta[:self._k_exog]
    # Autoregressive
    if self.order > 0:
        if np.any(self.switching_ar):
            for i in range(self.k_regimes):
                params[self.parameters[i, 'autoregressive']] = (
                        beta[self._k_exog:] * (i / self.k_regimes))
        else:
            params[self.parameters['autoregressive']] = beta[self._k_exog:]
    # Variance
    if self.switching_variance:
        params[self.parameters['variance']] = (
            np.linspace(variance / 10., variance, num=self.k_regimes))
    else:
        params[self.parameters['variance']] = variance

    return params

print('Parametros iniciales para Likelihood Evaluation:\n', mod_hamilton.
 ↪start_params)
```

```
Parametros iniciales para Likelihood Evaluation:
 [ 0.5         0.5         0.          0.27839396  0.96679597  0.30974497
  0.12725767 -0.12125847 -0.0892264 ]
```

Donde los dos primeros valores corresponden al **regimen Transition**, el tercer y cuarto termino vienen dados por la variable **exog**, el quinto valor viene dado por la **varianza** y el sexto valor en adelante es dado por el parametro **beta**. Lo principal del archivo MarkovSwitching_FAE, se encuentra dentro de la función .fit(), la cual se encarga de 'ajustar' el modelo. Para esto optimiza los valores iniciales para obtener 'Maximum likelihood estimation by scoring', junto con los algoritmos EM.

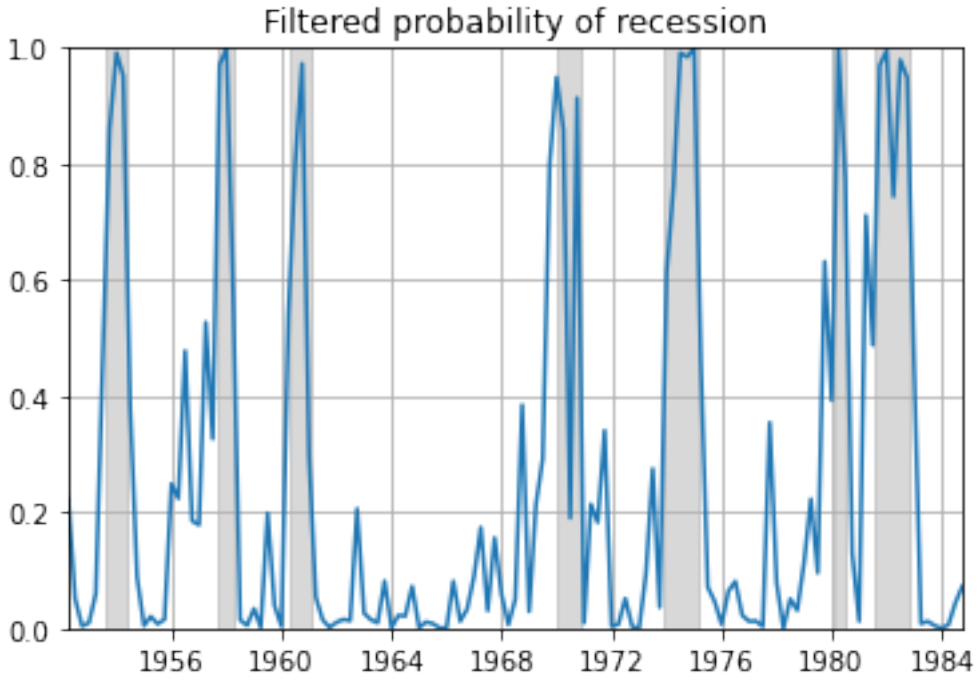
[9]:
```
res_hamilton = mod_hamilton.fit()
```

[10]:
```
print(res_hamilton.summary())
```

```
                        Markov Switching Model Results
==============================================================================
Dep. Variable:                          y   No. Observations:                  131
Model:              MarkovSwitching_FAE   Log Likelihood                -181.263
Date:                   Wed, 15 Apr 2020   AIC                            380.527
Time:                           02:41:56   BIC                            406.404
Sample:                       04-01-1951   HQIC                           391.042
                            - 10-01-1984
Covariance Type:                  approx
                            Regime 0 parameters
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.3588      0.265     -1.356      0.175      -0.877       0.160
                            Regime 1 parameters
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.1635      0.075     15.614      0.000       1.017       1.310
                          Non-switching parameters
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
sigma2         0.5914      0.103      5.761      0.000       0.390       0.793
ar.L1          0.0135      0.120      0.112      0.911      -0.222       0.249
ar.L2         -0.0575      0.138     -0.418      0.676      -0.327       0.212
ar.L3         -0.2470      0.107     -2.310      0.021      -0.457      -0.037
ar.L4         -0.2129      0.111     -1.926      0.054      -0.430       0.004
                        Regime transition parameters
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
p[0->0]        0.7547      0.097      7.819      0.000       0.565       0.944
p[1->0]        0.0959      0.038      2.542      0.011       0.022       0.170
==============================================================================

Warnings:
[1] Covariance matrix calculated using numerical (complex-step) differentiation.
```

[11]:
```
plt.plot(res_hamilton.filtered_marginal_probabilities[0])
plt.fill_between(usrec.index, 0, 1, where=usrec['USREC'].values, color='gray',␣
↪alpha=0.3)
plt.xlim(dta_hamilton.index[4], dta_hamilton.index[-1])
plt.ylim(0, 1)
plt.title('Filtered probability of recession')
plt.grid()
plt.show()
```



'Filtered' se refiere a una estimación de la probabilidad a tiempo t basado en la data incluyendo el tiempo t

[12]:
```
print(res_hamilton.expected_durations)
```

```
[ 4.0760479 10.4258927]
```

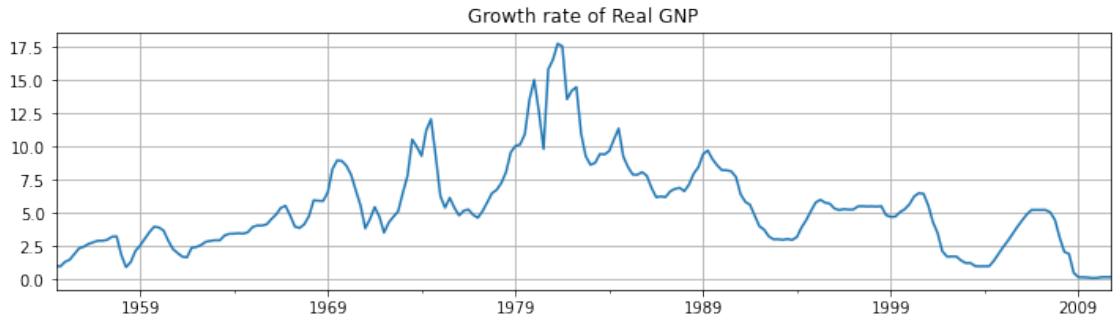
En este caso, se espera que una recesión dure aproximadamente un año (4 trimestres) y una expansión de aproximadamente dos años y medio.

De manera similar es posible obtener resultados para el FedFunds.

[13]:
```
from statsmodels.tsa.regime_switching.tests.test_markov_regression import␣
↪fedfunds

dta_fedfunds = pd.Series(fedfunds, index=pd.date_range('1954-07-01',␣
↪'2010-10-01', freq='QS'))
```

[14]:
```
# Plot the data
dta_fedfunds.plot(title='Growth rate of Real GNP', figsize=(12,3))
plt.grid()
```



[15]:
```
# Fit(). A switching mean is the default of the MarkovRegession model
mod_fedfunds = MarkovSwitching_FAE(dta_fedfunds, k_regimes=2, order=2)
res_fedfunds = mod_fedfunds.fit()
```

```
/home/juan/.local/lib/python3.6/site-packages/statsmodels/base/model.py:568:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)
```

[16]:
```
print(res_fedfunds.summary())
```

```
                        Markov Switching Model Results
==============================================================================
Dep. Variable:                          y   No. Observations:                  224
Model:             MarkovSwitching_FAE   Log Likelihood                -242.243
Date:                Wed, 15 Apr 2020   AIC                            502.485
Time:                        02:42:37   BIC                            533.190
Sample:                    07-01-1954   HQIC                           514.879
                         - 10-01-2010
Covariance Type:                approx
                          Regime 0 parameters
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          9.8110      1.167      8.410      0.000       7.525      12.098
ar.L1          0.8299      0.217      3.833      0.000       0.406       1.254
ar.L2          0.1700      0.236      0.719      0.472      -0.293       0.633
                          Regime 1 parameters
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.0244      0.878      5.725      0.000       3.304       6.744
ar.L1          1.4446      0.061     23.791      0.000       1.326       1.564
ar.L2         -0.5012      0.061     -8.248      0.000      -0.620      -0.382
                          Non-switching parameters
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
sigma2         0.4712      0.046     10.259      0.000       0.381       0.561
                        Regime transition parameters
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
p[0->0]        0.7482      0.215      3.482      0.000       0.327       1.169
p[1->0]        0.0045      0.004      1.002      0.316      -0.004       0.013
==============================================================================

Warnings:
[1] Covariance matrix calculated using numerical (complex-step) differentiation.
```