Description of the MS(k)-AR(p) Matlab package

Author: Marcelo Perlin / (PhD Student) / ICMA – Reading University

Email: marceloperlin@gmail.com / m.perlin@icmacentre.ac.uk

The purpose of this document is to report the models that this markov switching package for autoregressive models can handle. The mathematical notation behind the markov switching/Hamilton filter is not going to be exposed here since it is not the point. This definitely isn't an introductory document on the markov switching filter. My advice to those who are just starting out is to first read through the papers and books at references and then come back here to check the application of the algorithm and the general computational structure (input/output) of the model.

The Mean Equation

The autoregressive model that the Matlab function estimates is the following mean equation:

$$y_t = \alpha^{S_t} + \sum_{i=1}^{p} \beta_i^{S_t} y_{t-i} + \sigma^{S_t} \varepsilon_t$$

Where:

S_t represents the state at time *t*, that is, $S_t = 1....K$, where K is the number of states

p is the maximum lag at autoregressive component

α^{S_t} is the constant at state S_t

σ^{S_t} is the model's standard deviation at state S_t

$\beta_i^{S_t}$ is the beta coefficient for lag *i* at state S_t

ε_t is the residue which follows a particular distribution (in this case Normal or Student), with zero mean and std=var=1

As one can see, the code written in Matlab is related to the switching of the full mean equation. If you don't want to switch in the full equation, let's say for instance that you just want to switch in the alpha coefficient or any other setup, then you should look at my other Matlab submission (Estimation, Simulation and Forecasting of a Markov Switching Regression (General Case) in Matlab), where you can build your own econometric model (more details at the word document from the package's zip file).

As an example for the markov switching fitting function, the following options at MS_AR_Fit():

```
ar=4;                         % Number of lags in autoregressive component
k=2;                          % Number of states
x=ret;                        % Time series from example.mat
advOpt.distrib='Normal';      % Distribution assumption
advOpt.std_method='white';    % Method for standard error calculation
```

reefers to the estimation of the equations:

$$y_t = \alpha^{S_t=1} + \beta_1^{S_t=1} y_{t-1} + \beta_2^{S_t=1} y_{t-2} + \beta_3^{S_t=1} y_{t-3} + \beta_4^{S_t=1} y_{t-4} + \sigma^{S_t=1}\varepsilon_t$$

If $S_t = 1$ and

$$y_t = \alpha^{S_t=2} + \beta_1^{S_t=2} y_{t-1} + \beta_2^{S_t=2} y_{t-2} + \beta_3^{S_t=2} y_{t-3} + \beta_4^{S_t=2} y_{t-4} + \sigma^{S_t=2}\varepsilon_t$$

if $S_t = 2$

with:

$$P = \begin{bmatrix} p_{11} & p_{21} \\ p_{12} & p_{22} \end{bmatrix}$$

as the transition matrix, which controls the probability of a switch from state *j* (column *j*) to state *i* (row *i*). The sum of each column in *P* is equal to one, since they represent full probabilities of the process for each state.

If you run the script Example_MS_AR_Fit.m with matlab version[1] 7.5.0.342 (2007b), this is the output you should be getting if you have all the proper packages installed (garch, optimization, statistics). Notes that I've also added some explanations for the results (in red).

***** MS Optimizations terminated. *****

Final log Likelihood: 1292.7058	(This is the maximum log likelihood)
Number of parameters: 16	(number of parameters at model)
Distribution Assumption -> Normal	(distribution assumption used)
Method for standard error calculation -> white	(the method for calculation of standard errors ('white' for white(1984) or 'newey_west' for Newey and West (1987). Change it at advOpt.std_method)

-----> Final Parameters <-----

Parameters in State 1:

AR param -> 0.075871 -0.15138 0.032829 0.060858	(the autoregressive parameters at state 1)*
Std Errors -> 0.028974 0.013475 0.14594 0.011634	(associated standard errors)
Constant -> -5.2332e-005	(the constant (alpha) at mean equation, state 1)
Std Errors -> 9.7044e-005	(associated standard errors)
Std Dev -> 0.022611	(the standard deviation (sigma) at state 1)
Std Errors -> 0.0033469	(associated standard errors)

Parameters in State 2:

AR param -> 0.1655 -0.049969 -0.10448 0.0084654	(the autoregressive parameters at state 2)*
Std Errors -> 0.083387 0.080993 0.4814 0.0040488	(associated standard errors)
Constant -> 0.00019131	(the constant (alpha) at mean equation, state 2)
Std Errors -> 5.915e-005	(associated standard errors)
Std Dev -> 0.010982	(the standard deviation (sigma) at state 1)
Std Errors -> 0.014461	(associated standard errors)

-----> Transition Probabilities Matrix <----- (it controls the probability of a switch from state j (column j) to state *i* (row i).

0.983	0.026
0.016	0.973

* The AR parameters (the betas) are organized ascending order, meaning that, for the output given before, 0.075877 is the coefficient at lag 1 (state 1), and -0.049963 is the coefficient at lag 2 (state 2).

If this is not what you get at Matlab screen when running the example script, then something is seriously wrong with your Matlab. It's probably a versions problem (the .mat file doesn't work) and my suggestion is to avoid using the .mat files within the package and check if the main fitting routine still runs (you can do that with the simul and fit scripts).

[1] Different versions of matlab will output different numbers. The difference is quite small (insignificant) for the coefficients values, but increases significantly for the standard errors. For more information, please check the excel file within the zip file

The Distribution Equation (likelihood shape)

Notes that in the latest update I've added the choice for distribution assumption for the model (this is the dist parameter at the function). You can choose either the Normal distribution or the Student distribution (dist='Normal', dist='t').

The likelihood of the Normal distribution, for the case of generic state *j*, is given by:

$$L\left(y_t \mid \theta, S_t = j\right) = \frac{1}{\sigma^j \sqrt{2\pi}} \cdot \exp\left(-\frac{\left(y_t - \alpha^j - \sum_{i=1}^{p} \beta_i^j y_{t-i} \right)^2}{2\left(\sigma^j\right)^2} \right)$$

And the likelihood for the student distribution is:

$$L\left(y_t \mid \theta, S_t = j\right) = \frac{\Gamma\left(\frac{v^j + 1}{2}\right)}{\Gamma\left(\frac{v^j}{2}\right) * \sigma^j \sqrt{\pi v^j} \cdot \left(1 + \frac{\left(y_t - \alpha^j - \sum_{i=1}^{p} \beta_i^j y_{t-i} \right)^2}{v^j \left(\sigma^j\right)^2} \right)^{\left(\frac{v^j + 1}{2}\right)}}$$

Notes that for the *t* distribution there is an extra parameter (v^j) (degree of freedom) which, as the standard deviation, is also switching and is estimated from data. Those last equations are then used for maximum log likelihood estimation of the model as a constrained optimization problem (function fmincon()).

Using it for your own Data

You probably want to apply the package to you own series. This topic will set some advices of how you can do this if the model is not converging to a solution.

1. If you're dealing with returns (log or arithmetic), don't multiply them by 100 (just use them in the original form). I'm not exactly sure why, but this is something that people do when estimating in GAUSS and Ox and it is supposed to help the estimation. If you do this at Matlab, the fmincon function (the one which maximizes log likelihood) gets really messed up because the bounds of the parameters are badly shaped for this kind of issue, making the fmincon get stuck at –inf values. If you don't believe me, try to estimate the Example_Script_Fit with x=ret*100.

2. Try to make you explained series to behave around zero. For instance, if the model is not converging, then try diminishing the mean from the original time series. This will help the optimizing function (fmincon()) to find the solution. As you probably know, this is a gentle transformation.

3. Always try to estimate simple models. For instance, don't try to estimate any model with k>3 and ar>4. The model size (number of parameters) grows exponentially as k and ar grows. For instance, if k=5 and ar=4, then the model has 55 parameters to be estimated from data, which is definitely too much for fmincon(). Don't get me wrong, the package will try to estimate it, but the solution is probably a local maximum and you can't really trust the output you get. If you're still not convinced, then Google "parsimonious models" (go ahead, no one is watching ☺).

If, after those steps, you're still having problems converging to a solution, send a message to my email (**marceloperlin@gmail.com**) with a nice personal introduction and an attached zip file containing:

1) the scripts you're running (the main .m file)

2) the error message (if there is one) (could be in .txt or just in the email space)

3) the data (.xls, .txt or .mat)

and I'll take a look over it. I'll try to reply in less than 48 hours if the problem is simple (usually is).

Also, if you have any question regarding the package, feel free to contact me at my previously cited email.

Cheers.

Marcelo

References

DUEKER, M., NEELY, (2001) 'C., J., Can Markov Switching Models Predict Excess Foreign Exchange Returns?' *FRB of St. Louis Working Paper No. 2001-021F*. Available at SSRN: http://ssrn.com/abstract=648227.

HAMILTON, J., D. (2005) 'Regime Switching Models' *Palgrave Dictionary of Economic.*
KIM, C., J., NELSON, C., R. (1999) 'State Space Model with Regime Switching: Classical and Gibbs-Sampling Approaches with Applications' *The MIT press*.

NEWEY, B., WEST, D (1987) 'A Simple, Positive semidefinite, Heteroskedasticity and Autocorrelation Consistent Covariance Matrix' *Econometrica*, ed. 55, p. 347-370.

WHITE, H. (1984) 'Asymptotic Theory for Econometricians' New York: Academic Press.