An Implementation of Markov Regime Switching Model with Time Varying Transition Probabilities in Matlab

By Zhuanxin Ding, Ph.D.

First Version: June 12, 2012
This Version: June 22, 2012

Abstract

This memo explains how to use the MATLAB code for estimating a Markov Regime Switching Model with time varying transition probabilities. The code is developed by Zhuanxin Ding based on the original code by Marcelo Perlin for estimating a Markov Regime Switching Model with constant transition probability matrix.

1. Introduction

The Matlab code developed here (MS_Regress_tvtp) is based on the original Matlab code by Marcelo Perlin on Markov Regime Switching Model for constant transition probability matrix (MS_Regress). In order to understand the code here, please go through the original code and explanation by Marcelo Perlin.

Perlin's code is available at
https://sites.google.com/site/marceloperlin/matlab-code/ms_regress---a-package-for-markov-regime-switching-models-in-matlab

An explanation of Perlin's code can be found at
SSRN: http://ssrn.com/abstract=1714016 or http://dx.doi.org/10.2139/ssrn.1714016

The code structure, including subroutines and functions are almost the same. I attached a _tvtp for subroutine and function names used in this code. For example, Perlin's original code has a subroutine named MS_Regress_Fit.m, my corresponding subroutine will be named MS_Regress_Fit_tvtp.m. I would suggest the user build an m_files_tvtp directory in your computer to store the Matlab code here.

2. Major Differences with Perlin's Code

The major differences between this code and Perlin's code are as follows:

1) Implemented the Time Varying Transition Probability (TVTP) Regime Switching Model. Each regime transition probability can depend on (different) macro state variables. As in Perlin's code, this code can estimate a general TVTP model with more than 2 regimes.

2) The user can pre-specify his/her own initial values for parameters in estimation. It can be all the parameters, or a partial list of the parameters.
3) When the user wants to fix some parameters while estimate others, he/she does not need to provide the complete model specification for the field structure `advOpt.constConef`. Instead, one only needs to specify the relevant fields that need to be fixed.
4) In the current setting, the optimization problem becomes a pure non-linear optimization without constraints, so the Matlab unconstrained optimizer, *fminunc*, can also be used. However, the Matlab constrained optimizer, *fmincon*, is used as the default optimization engine. One can use `advOpt.optimizer='fminunc'` or `advOpt.optimizer='fmincon'` to specify which optimizer to use. The two optimizers will give slightly different optimal results by the nature of nonlinear optimization. It should also be noted that there is a bug in the Matlab 2012a version of *fminunc* that Matlab is trying to fix as of this writing.
5) The code can do a time series plot for mean, standard deviation, and transition probability if the input data structure is a time series object. `advOpt` needs to have a field named `YYYYMM` with year and month (e.g. 199912) as a vector.

3. Implementation

It is relatively easy to implement a 2-state Markov Regime Switching Model with Time Varying Transition Probability matrix. A GAUSS code is available upon request from Gabriel Perez-Quiros for implementing the 2-state Regime Switching Model discussed in Perez-Quiros and Timmermann (2000). The trick is to use the normal CDF function transformation on a linear model to get a probability estimation that is always between 0 and 1. However, when there are more than two states, it will not be so straightforward to constrain all the probabilities to be within 0 and 1 and to guarantee the sum of the probabilities to be 1 simultaneously.

In my implementation, I used a recursive time varying probability generating function for each probability cell when there are more than two states. Mathematically, for a *k*-state system, we have $(k-1)k$ independent time varying probability components that need to be estimated, i.e.,

$$Q_t = \begin{pmatrix} q_{11,t} & q_{12,t} & \cdots & q_{1k,t} \\ q_{21,t} & q_{22,t} & \cdots & q_{2k,t} \\ \vdots & \vdots & \ddots & \vdots \\ q_{k-1,1,t} & q_{k-1,2,t} & \cdots & q_{k-1,k,t} \\ 1 & 1 & \cdots & 1 \end{pmatrix} \tag{1}$$

For each probability cell (i, j) (where $i = 1,2,\cdots k-1,\ j = 1,2,\cdots k$) we specify a probability generating function as follows:

$$q_{ij,t} = \Phi(\mathbf{X}_{ij,t}\mathbf{b}_{ij}) \tag{2}$$

where $\Phi()$ is the cumulative normal density function, $\mathbf{X}_{ij,t}$ is the state variable vector for cell (i, j), and $\mathbf{b}_{ij}$ is the parameters to be estimated. The state variables can be different for different cells.

Next we generate an auxiliary matrix R_t based on Q_t as below

$$R_t = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1-q_{11,t} & 1-q_{12,t} & \cdots & 1-q_{1k,t} \\ \vdots & \vdots & \ddots & \vdots \\ \prod_{i=1}^{k-2}(1-q_{i1,t}) & \prod_{i=1}^{k-2}(1-q_{i2,t}) & \cdots & \prod_{i=1}^{k-2}(1-q_{ik,t}) \\ \prod_{i=1}^{k-1}(1-q_{i1,t}) & \prod_{i=1}^{k-1}(1-q_{i2,t}) & \cdots & \prod_{i=1}^{k-1}(1-q_{ik,t}) \end{pmatrix}. \tag{3}$$

The final time-varying transition probability matrix can be constructed as follows:

$$P_t = Q_t \bullet R_t = \begin{pmatrix} p_{11,t} & p_{12,t} & \cdots & p_{1k,t} \\ p_{21,t} & p_{22,t} & \cdots & p_{2k,t} \\ \vdots & \vdots & \ddots & \vdots \\ p_{k1,t} & p_{k2,t} & \cdots & p_{kk,t} \end{pmatrix} \tag{4}$$

where $\bullet$ stands for elementwise matrix product (Hadamard product).

The final probability transition matrix P_t can also be expressed as:

$$\begin{aligned} p_{1j,t} &= q_{1j,t} \\ p_{2j,t} &= (1-q_{1j,t})q_{2j,t} \\ &\vdots \\ p_{k-1,j,t} &= (1-q_{1j,t})(1-q_{2j,t})\cdots(1-q_{k-2,j,t})q_{k-1,j,t} \\ p_{kj,t} &= (1-q_{1j,t})(1-q_{2j,t})\cdots(1-q_{k-2,j,t})(1-q_{k-1,j,t}) \end{aligned} \tag{5}$$

for column j, $j = 1,2,\cdots k$. Each column will sum to 1 by design. It should be noted here that the row *i*, column *j* element of matrix P_t is the transition probability from state *j* to state *i*. This is consistent with the matrix notation (see Perlin 2012, p6) but is inverted from the conventional markov transition probability matrix notation (see Hamilton 1994, p679).

A consequence of the above specification is that we do not need to use the constrained optimization any more. So the model specification will be easier.

It should be emphasized that the estimated $q_{ij,t}$ is not the final probability; rather it's a component of the final probability.

4. Specification and Estimation

Compared with Perlin's original code, the only additional input is the macro state variables. The state variable is declared as a (*k*-1,*k*) cell `px`, each cell can contain a matrix of appropriate size for estimation. The specification is very general so that each probability generating function can depend on different macro state variables. The corresponding estimated coefficients for state variables will be in `Spec_Out.pa` and the time varying transition probability is in `Spec_Out.p`.

In Perlin's original code, one will need to specify the whole `advOpt.constCoeff` structure if he/she wants to fix some parameters. In the current code, the user only needs to specify the relevant parameter using an `advOpt.constCoeff0` structure. For example,

```
advOpt.constCoeff0.pa{1,2}={-1;'e'}
```

will fix the probability generating function for cell (1, 2) as follows:

$$q_{12,t} = \Phi(-1 + b_{12} x_{12,t}) \qquad (6)$$

where b_{12} will be estimated. One does not need to specify other parameter structures if they will be estimated.

Another feature of the current code is that the user can specify his/her own starting values for parameter estimation. This is especially useful if recursive estimation and backtesting are desired so that one can start from the initial values that are close to the optimal solution to save time. For example, the following specification

```
advOpt.Coeff0.pa{1,1}=[0.9; 0.1];
advOpt.Coeff0.pa{1,2}=[-0.7; 0.1];
```

will set the initial parameters for the transition probability equation at [0.9; 0.1] and [-0.7; 0.1] respectively.

5. Program Structure

The time varying transition probability Regime Switching code consists of the following subroutines:

1. MS_Regress_Fit_tvtp.m
2. checkInputs_tvtp.m
3. build_constCoeff_tvtp.m
4. check_constCoeff_tvtp.m
5. checkSize_constCoeff_tvtp.m
6. preCalc_MSModel_tvtp.m
7. MS_Regress_Lik_tvtp.m
8. getvarMatrix_MS_Regress_tvtp.m
9. param2spec_tvtp.m
10. spec2param_tvtp.m
11. MS_VAR_Fit_tvtp.m
12. confuneq_MS_Regress_tvtp.m
13. myMVNPDF_tvtp.m
14. mvnpdf_MSPERLIN_tvtp.m
15. doOutScreen_tvtp.m

16. doOutScreen_MSVAR_tvtp.m
17. doPlots_tvtp.m
18. tsPlots_tvtp.m
19. MS_Regress_For_tvtp.m
20. MS_Regress_Sim_tvtp.m.

It can be downloaded from Matlab file exchange at the following location:
http://www.mathworks.com/matlabcentral/fileexchange/37144

6. Citing the Package

If you have used the package for research and publications, make sure you cite both the original code by Perlin and this code, not just for acknowledgement but also for replication of results. For this memo, it should be cited as follows:

Ding, Zhuanxin, An Implementation of Markov Regime Switching Model with Time Varying Transition Probabilities in Matlab, (June 12, 2012). Available at SSRN: http://ssrn.com/abstract=2083332 or http://dx.doi.org/10.2139/ssrn.2083332

References

Hamilton, James D., Regime Switching Models. Palgrave Dictionary of Economics, 2005.

Hamilton, James D., Time Series Analysis. Princeton University Press, 1994.

Perez-Quiros, Gabriel, and Allan Timmermann, Firm Size and Cyclical Variations in Stock Returns, The Journal of Finance, Vol. LV, no. 3, June 2000, 1229-1262.

Perlin, Marcelo , MS_Regress - The MATLAB Package for Markov Regime Switching Models (May 7, 2012). Available at SSRN: http://ssrn.com/abstract=1714016 or http://dx.doi.org/10.2139/ssrn.1714016