ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 383-390

Estimation of Hybrid HMM/MLP models

Joseph Rynkiewicz

SAMOS, Université Paris I - Panthéon Sorbonne
Paris, France
rynkiewi@univ-paris1.fr

Abstract

Hybrid HMM/MLP models are useful to model piecewise stationary non-linear time series. A popular way to estimate the parameters of such models is to use the E.M. algorithm thanks to the Baum and Welch, forward-backward, algorithm. In this paper, we study a convenient way to estimate the parameters thanks to differential optimization. This new method can dramatically improve the time of calculus for long time series.

1 Introduction

Hidden Markov models have been introduced by Baum and Petrie [1] to deal with speech recognition. Decades later, Hamilton [3] applies generalization of this model to the American's GNP series. He uses indeed linear autoregressive models with Markov switching. In a previous paper [8], we show that the generalization of the Hamilton's model with MLP as regression function gives a model with good segmentation properties on the laser time series. The main way to estimate the parameters of these models is to use the Expectation-Maximization (E.M.) algorithm to find the maximum likelihood estimator. However, E.M. algorithm is an iterative algorithm and each M-step involves differential optimization of each MLP. These iterations inside other iterations are very expensive in CPU time. In this paper we show that careful parameterization of the model yields us a direct calculus of the derivative of the log-likelihood function (the score function). We can then estimate the model with classical differential optimization algorithms.

2 The model

Consider the process $(X_t, Y_t)_{t\in\mathbb{Z}}$, such that

1. $(X_t)_{t\in\mathbb{Z}}$ is a Markov chain in a finite state space $\mathbb{E} = \{e_1, ..., e_N\}$, which can be identified without loss of generality with the simplex of $\mathbb{R}^N$, where e_i are unit vectors in $\mathbb{R}^N$, with unity as the ith element and zeros elsewhere.

2. Given $(X_t)_{t\in\mathbb{Z}}$, the process $(Y_t)_{t\in\mathbb{Z}}$ is a sequence of non-linear autoregressive model in $\mathbb{R}^d$ and the distribution of Y_n depends only on X_n and $Y_{n-1}, \cdots, Y_{n-p}$, $p \in \mathbb{N}^*$.

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 383-390

Hence, for a fixed t , the dynamic of the model is :

$$Y_{t+1} = F_{X_{t+1}}(Y_{t-p+1}^{t}) + M_{X_{t+1}}\varepsilon_{t+1}$$

with $F_{X_{t+1}} \in \{F_{e_1}, ..., F_{e_N}\}$ non-linear functions represented by a MLP, $M_{X_{t+1}} \in \{M_{e_1}, ..., M_{e_N}\}$ invertible matrices and $(\varepsilon_t)_{t\in\mathbb{N}^*}$ a i.i.d sequence of Gaussian random variables of $\mathbb{R}^d$, $\mathcal{N}(0, I_d)$.

Write

$$a_{ij} = P(X_{t+1} = e_i \,|X_t = e_j) \text{ and } A = (a_{ij}) \in \mathbb{R}^{N\times N}$$

and define :

$$V_{t+1} := X_{t+1} - E[X_{t+1}\,|X_t] = X_{t+1} - AX_t.$$

With the previous notations, we obtain the general equations of the model, for $t \in \mathbb{N}$:

$$\begin{cases} X_{t+1} = AX_t + V_{t+1} \\ Y_{t+1} = F_{X_{t+1}}(Y_{t-p+1}^{t}) + M_{X_{t+1}}\varepsilon_{t+1} \end{cases} \tag{1}$$

This notation is similar to the notation used in Elliott [2].

3 Parameters of the model

3.1 Parameters of the transition matrix A

We suppose in the sequel that each element of A is strictly positive. The matrix A is stochastic, so the sum of a column is 1. There are $N-1$ free parameters for each column. Let $\upsilon_{ij} = \ln\frac{a_{ij}}{a_{Nj}}$, note that $\upsilon_{Nj} = 0$, and $(\upsilon_{1j}, \cdots, \upsilon_{N-1,j}) \in \mathbb{R}^{N-1}$.

Let A_j be the j-th column of A :

$$A_j = \left(\frac{e^{\upsilon_{ij}}}{1 + e^{\upsilon_{1j}} + \cdots + e^{\upsilon_{N-1j}}}\right)_{1\le i\le N}$$

It is easy to verify that the derivative of A with respect to parameter (υ_{ij}) is :

$$\frac{\partial a_{ij}}{\partial \upsilon_{ij}} = a_{ij}(1 - a_{ij}) \tag{2}$$

and for $l \neq i$

$$\frac{\partial a_{ij}}{\partial \upsilon_{lj}} = -a_{ij}a_{lj} \tag{3}$$

Moreover, if $k \neq j$ the derivative is null.

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 383-390

3.2 Parameters of the covariance matrix $\Sigma_{e_i} := M_{e_i} M_{e_i}^T$

The covariance matrix Σ_{e_i} is supposed to be positive definite, so we estimate his inverse $\Sigma_{e_i}^{-1}$. Note that this matrix is symmetric and we consider only the terms under the diagonal (diagonal included).

3.3 Parameters of MLP

We denote $(\omega_{e_i})_{1 \leq i \leq N}$ the parameters of MLP associated with state $(e_i)_{1 \leq i \leq N}$.

3.4 Notation of parameter vector of the model

The parameter θ is

$$\theta = (\omega_{e_i}^T, \cdots, \omega_{e_N}^T, \upsilon_{11}, \cdots, \upsilon_{(N-1)N}, (\Sigma_{e_1}^{-1})_{11}, \cdots, (\Sigma_{e_1}^{-1})_{dd}, \cdots, (\Sigma_{e_N}^{-1})_{dd})^T$$

where $(\Sigma_{e_i}^{-1})_{lk}$ is the coefficient of l-th row and k-th ($k \leq l$) column of matrix $\Sigma_{e_i}^{-1}$. $\omega_{e_i}^T$ is the weights vector of MLP F_{e_i} written in row.

4 Estimation of the parameters

We use the maximum likelihood estimator (MLE). There are two way of writing the likelihood, one has exponential complexity with respect to the observations (see Rabiner [7]), but we will use the second one which have only linear complexity with respect to the observations (see Hamilton [3]).

4.1 Calculus of the log-likelihood

Let $L_\theta(y_{-p+1}, \cdots, y_n)$ be the log-likelihood of observations $(y_{-p+1}, \cdots, y_n)$, we have

$$L_\theta(y_{-p+1}, ..., y_n) = L_\theta(y_n \,|y_{-p+1}, \cdots, y_{n-1}) \times \prod_{t=1}^{n-1} L_\theta(y_t \,|y_{-p+1}, \cdots, y_{t-1})$$

$$= \sum_{i=1}^{N} L_\theta(y_n \,|X_n = e_i, y_{-p+1}, \cdots, y_{n-1}) P_\theta(X_n = e_i \,|y_{-p+1}, \cdots, y_{n-1})$$

$$\times \prod_{t=1}^{n-1} L_\theta(y_t \,|y_{-p+1}, \cdots, y_{t-1})$$

Write in the sequel

- p_n^θ the vector with i-th component : $p_n^\theta(i) = P_\theta(X_n = e_i \,|y_{-p+1}, \cdots, y_{n-1})$, p_n^θ is known as the predictive filter of X_n.
- b_n^θ the vector with i-th component : $b_n^\theta(i) = L_\theta(y_n \,|X_n = e_i, y_{-p+1}, \cdots, y_{n-1})$, the conditional density of y_n knowing $X_n = e_i$ and $(y_{-p+1}, \cdots, y_{n-1})$.
- $B_n^\theta = diag\left(b_n^\theta\right)$ the matrix with b_n^θ for diagonal and zeros elsewhere.

The log-likelihood is then

$$\ln(L_\theta(y_1, \cdots, y_n)) = \sum_{t=1}^{n} \ln({b_t^\theta}^T p_t^\theta) \tag{4}$$

It is sufficient to calculate p_t^θ for $t = 1, \cdots, n$, in order to calculate the log-likelihood since :

$$b_t^\theta(i) = L_\theta(y_t \,|X_t = e_i, y_{t-1}, \cdots, y_{-p+1}) := \Phi_{e_i}(y_t - F_{e_i}(y_{t-p+1}^{t-1}))$$

where

$$\Phi_{e_i}(y_t - F_{e_i}(y_{t-1}))$$

is the conditional Gaussian density of y_t knowing $X_t = e_i$.

By Holst et al. [4], the predictive filter p_t^θ verifies the recurrence :

$$p_{t+1}^\theta = \frac{AB_t^\theta p_t^\theta}{{b_t^\theta}^T p_t^\theta} \tag{5}$$

We will suppose that the initial probability vector p_1^θ is the uniform distribution on E and we can recursively calculate p_t^θ, $t = 1, \cdots n$.

4.2 Derivative of the log-likelihood

Let θ_j be the j-th component of θ, we have :

$$\frac{\partial \ln(L_\theta(y_1, \cdots, y_n))}{\partial \theta_j} = \sum_{t=1}^{n} \frac{\frac{\partial {b_t^\theta}^T p_t^\theta}{\partial \theta_j}}{{b_t^\theta}^T p_t^\theta}$$

with

$$\frac{\partial {b_k^\theta}^T p_k^\theta}{\partial \theta_j} = \frac{\partial {b_k^\theta}^T}{\partial \theta_j} p_k^\theta + {b_k^\theta}^T \frac{\partial p_k^\theta}{\partial \theta_j} \tag{6}$$

After calculating the partial derivatives we obtain (see Rynkiewicz [9]):

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 383-390

$$\left\{\begin{array}{l}\frac{\partial b_k^\theta}{\partial \theta_j} = 0 \text{ if } \theta_j \text{ is a parameter of } A \\ \frac{\partial b_k^\theta}{\partial \left(\Sigma_{e_i}^{-1}\right)_{ll}} = b_k^\theta(i) \times \frac{1}{2}\left((\Sigma_{e_i})_{ll} - \left((y_k - F_{e_i}(y_{k-p}^{k-1}))(y_k - F_{e_i}(y_{k-p}^{k-1}))^T\right)_{ll}\right) \\ \frac{\partial b_k^\theta}{\partial \left(\Sigma_{e_i}^{-1}\right)_{l\neq m}} = b_k^\theta(i) \times \left((\Sigma_{e_i})_{lm} - \left((y_k - F_{e_i}(y_{k-p}^{k-1}))(y_k - F_{e_i}(y_{k-p}^{k-1}))^T\right)_{lm}\right) \\ \frac{\partial b_k^\theta}{\partial \theta_j} = b_k^\theta(i) \times \frac{1}{2}\sum_{1\leq m,l\leq d}(\Sigma_{e_i}^{-1})_{lm}\left(\left(F_{e_i}(y_{k-p}^{k-1}) - y_k\right)(l)\frac{\partial F_{e_i}(y_{k-p}^{k-1})(m)}{\partial \theta_j}\right. \\ \left. + \left(F_{e_i}(y_{k-p}^{k-1}) - y_k\right)(m)\frac{\partial F_{e_i}(y_{k-p}^{k-1})(l)}{\partial \theta_j}\right) \text{ if } \theta_j \text{ is a element of } \omega_{e_i}\end{array}\right.$$

and $\frac{\partial p_k^\theta}{\partial \theta_j}$ verifies the recurrence, with $\frac{\partial p_1^\theta}{\partial \theta_j} = 0$ for all j. :

$$\frac{\partial p_{k+1}^\theta}{\partial \theta_j} = \frac{A b_k^\theta}{b_k^{\theta\,T} p_k^\theta}\left[I - \frac{p_k^\theta b_k^{\theta\,T}}{b_k^{\theta\,T} p_k^\theta}\right]\frac{\partial p_k^\theta}{\partial \theta_j} + \left(\frac{\partial A}{\partial \theta_j} b_k^\theta + A\frac{\partial b_k^\theta}{\partial \theta_j}\right)\frac{p_k^\theta}{b_k^{\theta\,T} p_k^\theta} - \frac{A b_k^\theta p_k^\theta}{(b_k^{\theta\,T} p_k^\theta)^2}\left(\frac{\partial b_k^{\theta\,T}}{\partial \theta_j} p_k^\theta\right)$$

Moreover, if θ_j is an element of ω_{e_i} or $\Sigma_{e_i}^{-1}$

$$\frac{\partial B_k^\theta}{\partial \theta_j} = diag(0, \cdots, \frac{\partial b_k^\theta(i)}{\partial \theta_j}, \cdots, 0).$$

Finally

$$\left\{\begin{array}{lcll}\frac{\partial A}{\partial \theta_j} & = & C(\upsilon_{lm}) & \text{if } \theta_j \text{ is a parameter of } A,\ \theta_j = \upsilon_{lm} \\ \frac{\partial A}{\partial \theta_j} & = & O_{N\times N} & \text{else}\end{array}\right.$$

where $C(\upsilon_{lm})$ is defined by

$$\left\{\begin{array}{l} C_m(i) = -a_{im}a_{lm} \text{ if } i \neq l \\ C_m(i) = a_{lm}(1 - a_{lm}) \text{ if } i = l\end{array}\right.$$

5 Application : recursive estimation and performance on simulations

5.1 Recursive estimation

Since we can write the log-likelihood and it's derivative in a additive way, we can use classical method of recursive (or on-line) estimation of our model.

A recursive estimator θ_n of the parameter based on the first n observations of (y_t) can be written

$$\theta_{n+1} = \theta_n + \gamma_n H_n h_{n+1}$$

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 383-390

where γ_n is a gain sequence verifying

$$\gamma_n \leq 0, \sum_{n=1}^{\infty} \gamma_n = \infty, \sum_{n=1}^{\infty} \gamma_n^2 < \infty \tag{7}$$

and with h_n the score vector such that the j-th component is

$$h_n(j) = \frac{\partial b_n^\theta}{\partial \theta_n^j}^T p_n^\theta + {b_n^\theta}^T \frac{\partial p_n^\theta}{\partial \theta_n^j}$$

Note that the matrix H_n is an approximation of the inverse of the Fisher information matrix obtained by the Riccati lemma (see Rynkiewicz [9]).

The recursive estimation of such models is treated thanks an on-line E.M. algorithm in Holst et al. [4], but their numerical study is made on a very simple models (two regimes and two AR(1) for autoregressive models). Moreover, studies on classical HMM models show than recursive E.M. does not seem to be aa efficient way to estimate the parameters of HMM based models (see Krishnamurthy and Moore [5]). In the sequel, E.M. algorithm refers always to the off-line algorithm.

5.2 Performance on simulation

We simulate a series with two MLP with 2 entries, one hidden layer with 3 units (with hyperbolic tangents as activation function) and 2 output.

- The parameters for the two MLP are randomly chosen between -1 and 1.
- The transition matrix is

$$A = \begin{pmatrix} 0.95 & 0.1 \\ 0.05 & 0.9 \end{pmatrix}$$

- The covariance matrices are

$$\Sigma_1 = \begin{pmatrix} 0.29 & 0.34 \\ 0.34 & 1.48 \end{pmatrix} \text{ et } \Sigma_2 = \begin{pmatrix} 1.09 & 0.33 \\ 0.33 & 0.1 \end{pmatrix}$$

First we simulate a series with 1000 values. The CPU time for the estimation of parameter is obtained on a PC (400 MHz).

We randomly initialize 10 times each algorithm :

- E.M algorithm : The parameters are estimates with 50 iterations of E.M. algorithm and 10 iterations of BFGS optimization algorithm (see Press [6]) to optimize the weights of each MLP in the M-step.
- Differential optimization : The parameters are estimates with 100 iterations of BFGS.
- Recursive optimization : Since there are not enough data to ensure the convergence of the algorithm, we made 30 pass on the time series. The initial gain is 0.08 and the rule of decreasing is $\gamma_n = \frac{1}{n^{1/2+1e-16}}$.

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 383-390

The log-likelihood for the true parameter is -1.20, the final log-likelihood obtained for each algorithm and each initialization is :

E.M algorithm	Differential optimization	Recursive optimization
-1.36805	-2.13761	-1.30815
-1.40708	-1.69924	-1.25308
-2.4903	-1.40469	-1.33211
-1.72062	-1.53105	-1.34769
-1.35515	-1.69533	-1.33211
-1.76341	-1.67276	-1.35613
-1.48806	-1.56657	-1.2944
-1.56468	-1.75466	-1.24691
-1.5589	-1.53692	-1.3545
-1.53337	-1.56613	-1.37256
CPU :1365 s.	CPU :664.91 s.	CPU :222 s.

The estimated covariance matrices for the best model are

$$\widehat{\Sigma}_1 = \begin{pmatrix} 0.27 & 0.32 \\ 0.32 & 1.49 \end{pmatrix} \text{ and } \widehat{\Sigma}_2 = \begin{pmatrix} 1.10 & 0.33 \\ 0.33 & 0.10 \end{pmatrix}$$

and the estimated transition matrix is

$$\widehat{A} = \begin{pmatrix} 0.96 & 0.1 \\ 0.04 & 0.9 \end{pmatrix}$$

Note that both E.M. algorithm and differential optimization converge to a local maximum of the log-likelihood. There is no theoretical advantage to use one or another for reaching some better maxima. However the recursive estimation is, according a lot of empirical studies, more robust to achieve a good maximum.

We simulate now a long series (30000 values), we test only the recursive algorithm, because the other are too slow and will need many hours to converge.

The log-likelihood for the true parameter is : -1.18

Final log-likelihood
-1.20364
-1.19179
-1.20097
-1.22429
-1.20561
-1.18873
-1.22802
-1.29867
-1.2385
-1.18573
CPU : 248 s.

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 383-390

The estimated covariance matrix for best model are

$$\widehat{\Sigma}_1 = \begin{pmatrix} 0.29 & 0.35 \\ 0.35 & 1.51 \end{pmatrix} \text{ and } \widehat{\Sigma}_2 = \begin{pmatrix} 1.10 & 0.33 \\ 0.33 & 0.10 \end{pmatrix}$$

and the estimated covariance matrix is

$$\widehat{A} = \begin{pmatrix} 0.95 & 0.1 \\ 0.05 & 0.9 \end{pmatrix}$$

6 Conclusion

If we use the prediction filter of the state, the calculus of the log-likelihood of the Hybrid model has a linear complexity with respect to the observations. Moreover, if the model is parameterized with care, the calculus of the derivative of the log-likelihood is easy and we can use direct differential optimization to find the MLE. Finally, recursive algorithm deduced from this calculus yields us a very efficient method to estimate such models on long times series.

References

[1] L.E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite Markov chains. *Annals of Mathematical statistics*, 37:1559–1563, 1966.

[2] R. J. Elliott, L. Aggoun, and J. B. Moore. *Hidden Markov models : estimation and control*. Springer, 1997.

[3] J. D. Hamilton. A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica*, 57:357–384, 1989.

[4] U. Holst, G. Lindgren, J. Holst, and M. Thuvesholmen. Recursive estimation in Switching autoregressions with a Markov regime . *Journal of time series analysis*, 77:257–287, 1994.

[5] V. Krishnamurthy and John B. Moore. On-line estimation of hidden Markov model parameters based on the Kullback-Leibler information mesure. *IEEE transaction on signal processing*, 41:8:2557–2573, 1993.

[6] William H. Press, et al. *Numerical recipes in C : The art of scientific computing*. Cambridge University Press, 1992.

[7] L.R. Rabiner. A tutorial on hidden Markov models and selected application in speech application. *Proceedings of the IEEE*, 77:257–287, 1993.

[8] J. Rynkiewicz. Hybrid HMM/MLP models for time series prediction. In *ESANN'99*, 1999.

[9] J. Rynkiewicz. Modèles hybrides intégrant des réseaux de neurones artificiels à des modèles de chaînes de Markov cachées : applications à la prediction de séries temporelles. Thèse, Université de Paris 1, 2000.