IEICE Proceeding Series

Maximum likelihood estimation of quantized Gaussian autoregressive processes with Particle filters with resampling

András Horváth, Miklós Rásonyi

Vol. 1 pp. 427-430
Publication Date: 2014/03/17
Online ISSN: 2188-5079

2012 International Symposium on Nonlinear Theory and its Applications
NOLTA2012, Palma, Majorca, Spain, October 22-26, 2012

Maximum likelihood estimation of quantized Gaussian autoregressive processes with Particle filters with resampling

András Horváth† and Miklós Rásonyi‡

†Faculty of Information Technology, Pázmány Péter Catholic University
1444 Pf. 278, Budapest, Hungary
‡School of Mathematics, University of Edinburgh
James Clerk Maxwell Building, EH9 3JZ, Edinburgh, UK
Email: horvath.andras@itk.ppke.hu, Miklos.Rasonyi@ed.ac.uk

Abstract—In this paper we propose a method for the calculation of the maximum likelihood estimator for the autoregression coefficient of a stable quantized Gaussian autoregressive, AR(1) process. Our method uses particle filters with resampling and suits ideally on manycore architectures and can be implemented in a parallel way, this way yields fast processing speed. The extension to multidimensional autoregressivemoving-average (ARMA) systems is straightforward.

1. Introduction

Quantization is one of the most communly used nonlinear operation in communication [4]. The parameter estimation of quantized stochastic processes is used widely in practice. However this is a hard problem of mathematical statistics, the problem in theory can be solved [3].These theoretical solutions are computationaly expensive and in real-life problem e.g. speech recognition (speaker identification), only methods with relatively fast (real-time) processing speed can be applied. In this paper we investigate a scalable, paralellizable method that can estimate the parameter of an AR(1) process 10 times faster, than pervious methods.

We have selected the Gaussian ARMA processes, because there versatility makes them a natural class for investigating the effect of rounding off. An extension of our example to multidimensional (ARMA) systems is possible but it would lead to further complications which we prefer to avoid in the scope of this article.

2. Problem statement

We consider a stable Gaussian AR(1) process given by

$$X_n = \alpha_* X_{n-1} + \varepsilon_n, \quad n \geq 0,$$

where ε_n are i.i.d. random variables with law $N(0, \sigma_*^2)$ and $X_{-1} \in \mathbb{R}$ is a deterministic initial value. We assume that $X_{-1}, \sigma_* > 0$ are known. The autoregression parameter α_* is unknown, but it lies in the interval of admissible parameters $[\underline{\alpha}, \overline{\alpha}]$ where $-1 < \underline{\alpha} < \overline{\alpha} < 1$.

Only the rounded-off, quantized values $Y_n = q(X_n)$, $n \geq 0$ are observed, where the quantizer function q is defined by

$$q(x) = k, \text{ for } x \in [k - 1/2, k + 1/2), \ k \in \mathbf{Z}.$$

Our aim is to calculate the maximum likelihood (ML) estimator: $\hat{\alpha}_n$ for α_* based on the observation sequence $Y_n, n \geq 0$. This problem was investigated and already discussed in [5, 6, 7]. Following the footsteps of these authors, we will apply the expectation maximisation (EM) method. However, we will combine it with suitably designed particle filters (see sections 3, 4) while in [5, 6, 7] an MCMC approach was used instead. We analyse our simulation results in section 5.

3. Motivation for using particle filters

Particle filters (PFs) can be and are used for state- and probability estimations in numerous applications. They work well with difficult and non-linear Markovian models where the Kalman filters can not be used. Particle filters without resampling proved not to be efficient enough for complex, practical problems. Particle filters with resampling have much better performance. They lose, however, the property of being suitable for estimating functionals depending on the entire trajectory, because we alter the distribution of the particles during resampling since this would require keeping alive all the particles along the whole trajectory.

Nevertheless, in the following sections we would like to show how particle filters with resampling can be used succesfully as part of the EM method for parameter estimation of quantized AR processes. To have simple formulas, we assume $X_{-1} = 0$ and $\sigma_* = 1/2$, but the conclusions hold for arbitrary values of these parameters.

For the moment we fix n, the number of iterations and concentrate on determining $\hat{\alpha}_n$. Introducing the notation

$$p(\alpha; y_0, \ldots, y_n) = \frac{1}{\pi^{(n+1)/2}} e^{-y_0^2 - \sum_{j=1}^n (y_j - \alpha y_{j-1})^2},$$

the estimate $\hat{\alpha}_n$ can be found by maximising

$$e^{L_n(\alpha)} = \int_{Q(Y_0,\ldots,Y_n)} p(\alpha; y_0, \ldots, y_n) dy_0 \ldots dy_n, \qquad (1)$$

in α which amounts, by taking the derivative, to solving

$$\int_{Q(Y_0,\ldots,Y_n)} \left(\sum_{j=1}^{n} -y_{j-1}(y_j - \alpha y_{j-1}) \right) p(\alpha; y_0, \ldots, y_n) dy_0 \ldots dy_n = 0, \tag{2}$$

here $Q(Y_0, \ldots, Y_n)$ denotes the cube $[Y_0 - 1/2, Y_0 + 1/2] \times \ldots \times [Y_n - 1/2, Y_n + 1/2]$.

Since equation 2 is a complicated nonlinear equation, following [5, 6, 7], we apply the EM method when implementing the calculation of the ML estimate. In the present context this means setting an arbitrary value $\tilde{\alpha}_0$ and then determining $\tilde{\alpha}_l$ recursively as the root of

$$\int_{Q(Y_0,\ldots,Y_n)} \left(\sum_{j=1}^{n} -y_{j-1}(y_j - \alpha y_{j-1}) \right) p(\tilde{\alpha}_{l-1}; y_0, \ldots, y_n) dy_0 \ldots dy_n = 0 \tag{3}$$

Based on empirical evidence and theoretical results in various model classes one expects that, after a suitable number of iterations (i.e. l large enough), one can get close to $\hat{\alpha}_n$. The great advantage of equation 3 is that it is linear in α, this way its solution can be obtained easily by the usage of simple gradient methods.

Calculating $\tilde{\alpha}_l$ still poses a significant challenge as it requires the estimate of a functional of a trajectory (the integral in equation 3). Unlike in [5, 6, 7], where a Markov chain Monte Carlo approach is suggested, we apply particle filters for the determination of the integrals 3. Note also that in [5, 6, 7] test runs concentrated on the i.i.d. case (where $\alpha_* = 0$ is known) and the (unknown) expectation of ε_1 was estimated. In the present paper we took $E\varepsilon_1 = 0$ for simplicity but α_* is non-zero which leads to a far more difficult, non-i.i.d. dynamics for the processes X_n, Y_n and hence poses a much more complex problem.

Applying PFs without resampling seems hopeless, especially considering the fact that in the present case we have a non-discrete, infinite state space where probability distribution of the trajectories is continuos so even by discretization without resampling we need an extremely large number of particles to simulate these trajectories.

It turns out that, with a small alteration PFs with resampling (i.e. the standard bootstrap filter) can be succesfully used for probability estimation. The changes in the algorithm require some extra computation, but the number of particles can be decreased drastically compared to the method without resampling.

4. Estimation of functionals of a trajectory by Particle Filters with resampling

We explain a method to evaluate the integral in equation 1 (i.e. equation 4 below), calculation of the integral in EQ(3) can be carried out much in the same way.

In practical problems we usually have a given trajectory y_t, $t = 0, 1, \ldots, T$ generated by the previous model with T iterations (in our paper we chose $T = 100$ or $T = 200$).

Our aim is to estimate the probability that from the given model we will get the given trajectory of observations, i.e.:

$$P(Y_t = y_t, Y_{t-1} = y_{t-1} \ldots Y_0 = y_0). \tag{4}$$

Sequential Monte Carlo methods such as particle filters are routinely used in the case of such complex, nonlinear stochastic models with non-linear observations. The present system, however, has been revealed to be more challenging than usual.

First we look at PFs without resampling. We start off with an initial guess $\tilde{\alpha}_0$ for $\hat{\alpha_n}$.

At the initial step ($t = 0$) we generate N particles, ξ_k^0, $k = 1, \ldots, N$, i.i.d. with the same distribution $N(X_{-1}, \sigma_*^2)$ (i.e. with the law of $X_0^{(\tilde{\alpha}_0)}$.

At step t we iterate the particles using the system dynamics, i.e.

$$\xi_k^{t+1} := \tilde{\alpha}_0 \xi_k^t + \epsilon_{t+1}^k$$

with ϵ_{t+1}^k i.i.d. Gaussian with mean 0 and variance σ_*.

Every particle represents one possible trajectory and after the last (Tth) iteration we can calculate how many particles are identical with our observation. Based on this amount our estimate for the probability in EQ(4) is the relative frequency

$$p(\tilde{\alpha}_{l-1}; y_0, \ldots, y_n) = \frac{\sum_{i=1}^{N} I_{y_0 \ldots y_n}(\xi_i^0 \ldots \xi_i^n)}{N}.$$

Where the nominator on the right hand side represents the number of particles which results the same observations for every time and the denominator N is the total number of particles used. In theory this fraction converges to the real value in EQ 4, however in practice, especially in case of long, multidimensional trajectories we have to use an extremely large number of particles, because after each step the number of matching trajectories will always be decreased by the same amount in average.

However, PFs with resampling (based on the current observation) are more promising, especially when combined with importance sampling. We can decrease the number of useless particles, and increase the number of useful ones. The main point is that, in this way, the number of particles required will not depend on the length of the trajectory. PFs with resampling work well for estimating the density of X_T conditional to the observations $Y_0, \ldots, Y_T$. But during the resampling procedure the ratio between the number of good and bad particles gets lost and the estimation of the probability for the whole observed trajectory does not work in the obvious manner.

Here we briefly describe, how one can calculate the probabilities between state transitions: although at each step certain particles do not match the observation and hence drop out from the cohort (as they are resampled with probability 0) one can 'blow up' the remaining cohort of, say, M 'matching particles', respecting the overall distribution of particles, to size N again. Getting more into the technicalities, this involves a discretization of the state

space $[Y_t - 1/2, Y_t + 1/2)$ using a suitable mesh size; then one has to calculate how many of the matching particles fall into the respective intervals and from this one can create a new cohort of N following the same distribution as the previous M particles.

We can approximate the distribution of these particles by dividing the set of the possible observations into Q intervals and calculate the number of matching particles (ξ_k^*) in every interval.

$$f(q) = \frac{\sum_k \xi_k^* eI_q}{N} \forall q = L_1 : Q \tag{5}$$

the limit of $f(q)$ is the distribution of the hidden states as the number of particles approaches infinity.

Let us note the approximated distribution of the particles after the resampling with $g(i)$. There are no known theoretical connection between $f(i)$ and $g(i)$. $g(i)$ can be calculated easily in the same way as in equation 5.

Our aim is now to reserve the distribution of the particles. We can introduce the following operator, which is how the resampling step alters the distribution of the cohort:

$$\nu(f(q)) = g(q) \tag{6}$$

That transform $f(q)$ into $g(q)$. It can be easily seen that ν is always invertable and we can use ν^{-1} to restore $f(q)$ from $g(q)$.

All we have to do is to set the weights in interval q to $\nu^{-1}(g(q))$ after this step the weighted sum in interval q will be equal to the probability that the hidden state will be an element of interval q.

$$f(q) = \sum_k w_k^t \xi_k^t : \xi_k^* eI_q \tag{7}$$

where ξ_k^t are particles after the resampling step and w_k^t represents the weight of the particle.

The mesh size (Q) is another parameter to be optimized for concrete test runs. An optimal mesh size can be calculated/approximated offline for every known model. We aslo have to note, when the mesh size is: $Q = 1$, we will have the simple method with resampling,the regular bootstrap filter, where we do not alter the weights of the cohort. We do not have to alter bootstrap filters to expand the number of good particles after a resampling (because this is what resampling does), if all the weights are zero outside the given interval, only N particles within the interval can be selected, however some particles will be selected repeatedly and they will be overrepresented in the current distribution (it is also possible that some particles will not be selected- they will be underrepresented). We will not alter the distribution of the particle, we will alter the weights and with this the distribution that the particles are forming. This way the distribution of the particles is the same as in case of the regular boostrap filter, which convergence has been proofed [2].

In this way we can calculate probabilities of trajectories as well as integrals of EQ(3) using PFs with resampling, which makes the number of particles to be used exponentially less.

From this point on we follow the EM method: having calculated the integral in EQ(3) using $\tilde{\alpha}_0$ as a parameter, one may get $\tilde{\alpha}_1$ solving EQ(3), then iterate the procedure to get better and better approximations $\tilde{\alpha}_l$ of $\hat{\alpha}_n$, where l is the number of iterations performed.

This way we can melt the advantages of probability calculations with the fast, parallel computation possibility of bootstrap filters.

5. Simulations and Results

We have created a virtual machine to test the usability of our method in practice. We have simulated different AR(1) processes and we have tried to estimate the parameters according to the previously described algorithm. We have repeated every measurement 10.000 times and calculated the absolute error as an average to eliminate the noise of our measurements.

As it can be seen in table 1 and 2 our result have the same accuracy as in ... It can also be seen, that the usage of importance-sampling results a much lower error, especially in case of large number of particles. This low errors (we can approximate α with error ± 0.001) is accurate enough to be used in case of real life problems.

With this type of calculation one iteration of $\tilde{\alpha}_l$ with 1000 particles and over a 100 steps long trajectory could be calculated in 4.6 seconds on a single core architecture. Even on a four core architecture, the execution speed can be further decreased: the same estimation can be calculated in 1.8 sec on an Intel Q9600 CPU.

The main advantage of our method is, that almost every step of the algorithm (including the time consuming resampling step) is paralellizable, hence on a proper architecture such as an FPGA, GPU or digital-CNN architecture the computation speed could be further decreased, see [1].

In case of the method without resampling, $\tilde{\alpha}_l$ can not be estimated with 1000 particles at all, because in most of the cases there are no trajectories that will match our observations. For the current model and for a trajectory of length 100 the PF with resampling could calculate the probability with 300 particles with the same accuracy as a PF without resampling with 10.000.000 particles. This shows how drastically decrasing the number of efficiently used particles without reasmpling. While we keep all of the particles and the computational power efficient, for a longer trajectory the computation time will increase linearly (instead of the exponential growth of the normal method), because the number of the efficient particles are not decreasing.

The bigest problem in case of this algorithm is, that we have no information about α. If the distance between the real value and the distance between our initial value is too large there will not be any matching particles to the observation. This way our distribution will be a constant zero function, which is not a real distribution. To avoid this we

can restart our algorithm with an other parameter, however this takes an extra iteration. To avoid this extra calculation and further speed up our algorithm we can extend our model and add $\tilde{\alpha}_0$ as a parameter, this way we can start our algorithm with multiple values of $\tilde{\alpha}$ and the parameters not matching the observations will be filtered out during the iterations by the resampling step.

6. Conclusion

In this paper we have described a method that melts the advantages of probability calculations with the fast, parallel computation possibility of bootstrap filters. The processing speed of our method depends linearly on the trajectory length and its accuracy increasing with it. In previous methods and solutions either the accuracy did no change without the exponential increase of computation or the runing time depended exponentialy on the length of the trajectory. Our method fits ideally on a multicore architecture such as an FPGA, GPU or digital-CNN and with this we could create a device that can solve this difficult problem in real-time with low power consumption.

Acknowledgments

The support of the grants TÁMOP-4.2.1.B-11/2/KMR-2011-0002 and TÁMOP-4.2.2/B-10/1-2010-0014 are gratefully acknowledged.

References

[1] A. Horvath, M. Rasonyi Topographic Implementation of Particle Filters on Cellular Processor Arrays. *ELSEVIER Signal Processing Submitted*

[2] D. Crisan, A. Doucet. A Survey of Convergence Results on Particle Filtering Methods for Practitioners. *IEEE Transactions on Signal Processing* 736-746, 2002

[3] D. Magill. Optimal adaptive estimation of sampled stochastic processes. *IEEE Transactions on Automatic Control* 434 - 439, 1965

[4] H. Boche, U. J. Monich. Behavior of the Quantization Operator for Bandlimited, Nonoversampled Signals. *IEEE Transactions on Information Theory* 2433 - 2440, 2010

[5] L. Finesso, L. Gerencsér, I. Kmecs. A randomized EM-algorithm for estimating quantized linear Gaussian regression. *Proceedings of the 38th IEEE Conference on Decision and Control, Phoenix, Arizona, USA*. 5100–5101, 1999.

[6] L. Finesso, L. Gerencsér, I. Kmecs. A recursive randomized EM-algorithm for estimation under quantization error. *Proceedings of the American Control Conference, Chicago, Illinois.*, 790–791, 2000.

[7] L. Gerencsér, I. Kmecs, B. Torma. Quantization with adaptation – estimation of Gaussian linear models. *Communications in Information and Systems*, **8**, 223–244, 2008.

7. Tables

NumP	1000	1000	1000	2000
Div	10	20	50	20
TrajL	100	100	100	500
I1	0.1472	0.0682	0.0672	0.0141
I2	0.1263	0.0680	0.0672	0.0125
I3	0.1080	0.0678	0.0672	0.0114
I4	0.0860	0.0675	0.0669	0.0112
I5	0.0750	0.0673	0.0669	0.0110
I6	0.0724	0.0672	0.0666	0.0110
I7	0.0722	0.0666	0.0665	0.0110
I8	0.0719	0.0666	0.0658	0.0110

Table 1: The first row contains the number of particles (N), the second row contains 'how fine' the interval of the observation was divided (Q) for the distribution estimation, The third row is the length of the trajectory (n) and the rows behind this are the errors to every iteration of the EM method. This experiments were done with using the importance-sampling method

NumP	1000	1000	1000	1000
Div	10	20	50	100
TrajL	100	100	100	100
I1	0.0847	0.1152	0.1661	0.2127
I2	0.0826	0.0991	0.1524	0.2056
I3	0.0801	0.0976	0.1471	0.1948
I4	0.0760	0.0958	0.1439	0.1866
I5	0.0750	0.0951	0.1393	0.1820
I6	0.0741	0.0948	0.1373	0.1748
I7	0.0722	0.0910	0.1274	0.1647
I8	0.0719	0.0908	0.1268	0.1629

Table 2: The first row contains the number of particles (N), the second row contains 'how fine' the interval of the observation was divided (Q) for the distribution estimation, The third row is the length of the trajectory (n) and the rows behind this are the errors to every iteration of the EM method. This experiments were done without using the importance-sampling method