UPPSALA UNIVERSITET
INSTITUTIONEN FÖR INFORMATIONSTEKNOLOGI

Recursive identification

Abstract

This note presents the basic recursive least squares (RLS) algorithm for recursively estimating parameters in linear regression models. In order to track time varying parameters a forgetting factor and a Kalman filter method are described. These methods may be used in some of the extra tasks in the project and is also used in the compulsory computer exercise (beräkningslaboration) No 5 .

Bengt Carlsson

Sept 20, 2011

Innehåll

2 Introduction

Recursive identification is the name for estimation algorithms where the estimated parameters are updated for each new observation. The need for recursive identification arises when we want to have a new parameter estimate after each new observation. Recursive identification relies on fast algorithms where the computational burden and required memory do not increase with time. The key idea is to arrange the computation so that the estimate obtained for *N* observations is used in order to calculate the estimate for *N+1* observations. Typically, the new estimate is equal to the previous estimate plus a correction term which depends on the some gain times the prediction error.

Recursive identification is also the key tool for handling the (common) case when the system dynamics are varying over time. As will be shown, by simple modification the basic algorithm can be changed so it can track time varying dynamics.

3 Recursive algorithms

3.1 Recursive identification of a constant

Consider the simple case when we want to estimate the mean value of a time series. The following simple model (predictor) is then used

$$\hat{y}(t) = m = \varphi^T(t)\theta$$

where φ=1, and θ=m is a scalar. Given the observations y(1), y(2)… y(t) the least squares solution is the arithmetic mean (see the course material of "Linear regression")

$$\hat{\theta}(t) = \hat{m}(t) = \frac{1}{t}\sum_{s=1}^{t} y(s) \qquad (1)$$

We can easily reformulate this expression to a recursive algorithm:

$$\hat{\theta}(t) = \frac{1}{t}\sum_{s=1}^{t} y(s) = \frac{1}{t}\left[\sum_{s=1}^{t-1} y(s) + y(t)\right] = \frac{1}{t}\left[(t-1)\hat{\theta}(t-1) + y(t)\right]$$
$$= \hat{\theta}(t-1) + \frac{1}{t}\left[y(t) - \hat{\theta}(t-1)\right] \qquad (2)$$

In the third equality we have used $\hat{\theta}(t-1) = \frac{1}{t-1}\sum_{s=1}^{t-1} y(s)$. Note by this reformulation we have that the new estimate (at time *t)* is equal to the old estimate (the estimate at time *t-1*) plus a correction term which depends on a gain (*1/t*) and the prediction error using the estimate at time *t-1*. To use (2) we need only to store the latest estimate (that is no old measurements need to be stored). With $\hat{\theta}(0) = 0$, the estimate (2) gives an identical estimate as the "off-line" estimate (1). However, a priori information about θcan be used for the choice of $\hat{\theta}(0)$, see also Section 3.5.

Finally, observe that the gain sequence *1/t* goes to zero as *t* increases. After some time, the estimate will note change very much even if the true mean value changes. That makes the algorithm not suitable for tracking time variations in the mean value. A remedy to this is presented in Section 3.3.

3.2 The recursive least square method

We will next present a recursive way to estimate the parameters in a linear regression model

$$\hat{y}(t) = \varphi^T(t)\theta \qquad (3)$$

where $\varphi^T(t)$ is the regressions vector and θ is a vector of unknown parameters. The least squares estimate (given data up to time t) is given by (see the course material of “Linear regression”)

$$\hat{\theta}(t) = [\sum_{s=1}^{t} \varphi(s)\varphi^T(s)]^{-1} \sum_{s=1}^{t} \varphi(s)y(s) \qquad (4)$$

In Appendix (1) it is shown that off-line estimate (4) can be written recursively as

$$\begin{aligned} \hat{\theta}(t) &= \hat{\theta}(t-1) + P(t)\varphi(t)\left(y(t) - \varphi^T(t)\hat{\theta}(t-1)\right) \\ P(t) &= P(t-1) - \frac{P(t-1)\varphi(t)\varphi^T(t)P(t-1)}{1+\varphi^T(t)P(t-1)\varphi(t)} \end{aligned} \qquad (5)$$

The algorithm (5) is known as RLS (Recursive Least Squares). The algorithm requires very little memory and the computational complexity is low (note that no matrix inversion is needed). For every new sample, only the matrix P needs to be updated.

To start up the algorithm, *P(0)* and $\hat{\theta}(0)$ need to be set by the user, see Section 3.5.

Note that the term $y(t) - \varphi^T(t)\hat{\theta}(t-1)$ is the prediction error where $\varphi^T(t)\hat{\theta}(t-1)$ is the prediction of y(t) using the model available at time *t-1*. If the prediction error is small, only a small update is made, which is reasonable.

The algorithm (5) has the same basic structure as (3). It can be shown that P(t) goes to zero as *t* goes to infinity, hence the algorithm is not suitable for tracking time varying system.

Remark: We can write the gain sequence as

$$P(t)\varphi(t) = K(t) = \frac{P(t-1)\varphi(t)}{1+\varphi^T(t)P(t-1)\varphi(t)}$$

3.3 RLS with forgetting factor

The RLS algorithm (5) (or the off-line method (4)) is suitable if it is known that the system dynamics is constant. However, if the system is given by

$$y(t) = \varphi^T(t)\theta_o(t) + e(t) \qquad (6)$$

where $\theta_o(t)$ describes a time varying parameter vector, the RLS method is unsuitable.

One way to obtain an estimation algorithm which can track parameter changes is to change the least squares criterion to

$$J = \sum_{s=1}^{t} \lambda^{t-s} \; (y(s) - \hat{y}(s))^2 \tag{7}$$

In the criterion, a so called forgetting factor λ (0< $\lambda \leq 1$) is introduced. A forgetting factor less than one weights old prediction errors less than new ones (for $\lambda = 1$, the standard least squares criterion is retrieved). The lower the forgetting factor is, the faster old prediction errors will be "forgotten" (that is, not used in determine the estimate) .

The modified RLS (also known as the discounted RLS) based on the criterion (6) is given by (a proof is given in Appendix 2):

$$\begin{cases} \hat{\theta}(t) = \hat{\theta}(t-1) + P(t)\varphi(t)\left(y(t) - \varphi^T(t)\hat{\theta}(t-1)\right) \\ \\ \qquad P(t) = \frac{1}{\lambda}\left(P(t-1) - \frac{P(t-1)\varphi(t)\varphi^T(t)P(t-1)}{\lambda + \varphi^T(t)P(t-1)\varphi(t)}\right) \end{cases} \tag{8}$$

For $\lambda < 1$, the gain matrix P(t) will not converge to zero as time goes to infinity, hence the algorithm will be able to update the parameters when the system is time varying. In practice, λ, is a design parameter, a low value of λ gives a fast tracking of time varying parameters but a high noise sensitivity. A typically choice of λ is in the range 0.94-0.999.

3.4 A Kalman filter approach[1]

Another approach to estimate time varying parameters is to extend the system description so that we explicitly model the expected variation. This can be done with the following system:

$$\begin{aligned} \theta(t+1) &= \theta(t) + v(t) \\ y(t) &= \varphi^T(t)\theta(t) + e(t) \end{aligned} \tag{9}$$

where $v(t)$ is (vector valued) white noise process with covariance matrix R_1. The variations in the θ-vector is modeled as a "random walk". The measurement noise $e(t)$ is assumed white with variance R_2. The description (9) can be compared with a standard discrete-time state space model (where θ in (9) corresponds to the state vector *x* in the state space model). For state space models it is known that the Kalman filter gives (under some assumptions) the best estimate of the unknown states. This also holds for (9). Applying a Kalman filter to (9 gives

$$\begin{aligned} \hat{\theta}(t) &= \hat{\theta}(t-1) + K(t)\left[y(t) - \varphi^T(t)\hat{\theta}(t-1)\right] \\ K(t) &= \frac{P(t-1)\varphi(t)}{1 + \varphi^T(t)P(t-1)\varphi(t)} \\ P(t) &= P(t-1) - \frac{P(t-1)\varphi(t)\varphi^T(t)P(t-1)}{1 + \varphi^T(t)P(t-1)\varphi(t)} + R_1 \end{aligned} \tag{10}$$

[1] This section requires the course "Datorbaserad styrning/Reglerteknisk design" and may be omitted by students not having read that course. The exam will not contain any theory from this section.

In (10) we have set R_2 =1, since only the ratio $\frac{R_1}{R_2}$ affect the gain $K(t)$. In practice the matrix R_1 is normally used as a design/tuning parameter. Typically R_1 is chosen as a diagonal matrix where each diagonal element reflects the size of the corresponding parameter variation. The Kalman filter is particular suitable when it is a priori known that some parameters are changing significantly faster than others. Assume for example that our model has two parameters, and that we know that the first parameter is not changing. A typical choice of R_1 is then

$$R_1 = \begin{bmatrix} 0 & 0 \\ 0 & r \end{bmatrix}$$

where we now only have one tuning parameter *r*. For problems like this, the Kalman filter will work better than the RLS with forgetting factor. The reason is that in RLS with forgetting factor we have no way to use information on how different parameters are varying.

In the Kalman filter, the same trade off between good tracking ability and low noise sensitivity as in the RLS with forgetting factor has to be made.

3.5 Initial values

In order to start up a recursive algorithm the initial values *P(0)* and $\hat{\theta}(0)$ must be specified. Also the $\varphi(t)$-vector needs to be filled with values (typically one starts the recursive algorithm at a time when the $\varphi(t)$-vector can be filled with measured values of old y and u).

Without any knowledge of the true parameters it is common practice to use

$$\hat{\theta}(0) = 0 \text{ (a vector) and } P(0) = \text{cI}$$

where *I* is the identity matrix and c is a large number (say 1000).

If we have some knowledge about the true system parameter vector this should be used in $\hat{\theta}(0)$. The choice of *P(0)* should reflect our confidence in our initial guess $\widehat{\theta}(0)$, the more we believe in our initial estimate the lower we should set *P(0)***.** Note that if *P(0)* is small then *K(t)* will be small for all *t* and the estimate will not jump far away from $\widehat{\theta}(0)$.

4 Numerical illustrations

4.1 Recursive identification of a time varying parameter

We will here illustrate RLS with forgetting factor for the simple case when the data is given by

$$y(t) = m(t) + e(t)$$

where m(t)=0 for t<200 and m(t)=5 for t>200, see the upper plot in Figure 1.

In the lower plot of Figure 1, we see the results from RLS for two different forgetting factors $\lambda = 0.98$ and, $\lambda = 0.90$. As seen from the figure, a low forgetting factor gives a fast tracking but a noisy estimate compared to when a forgetting factor closed to one is used.

This basic trade-off between fast tracking and noise sensitivity holds for all estimation schemes.

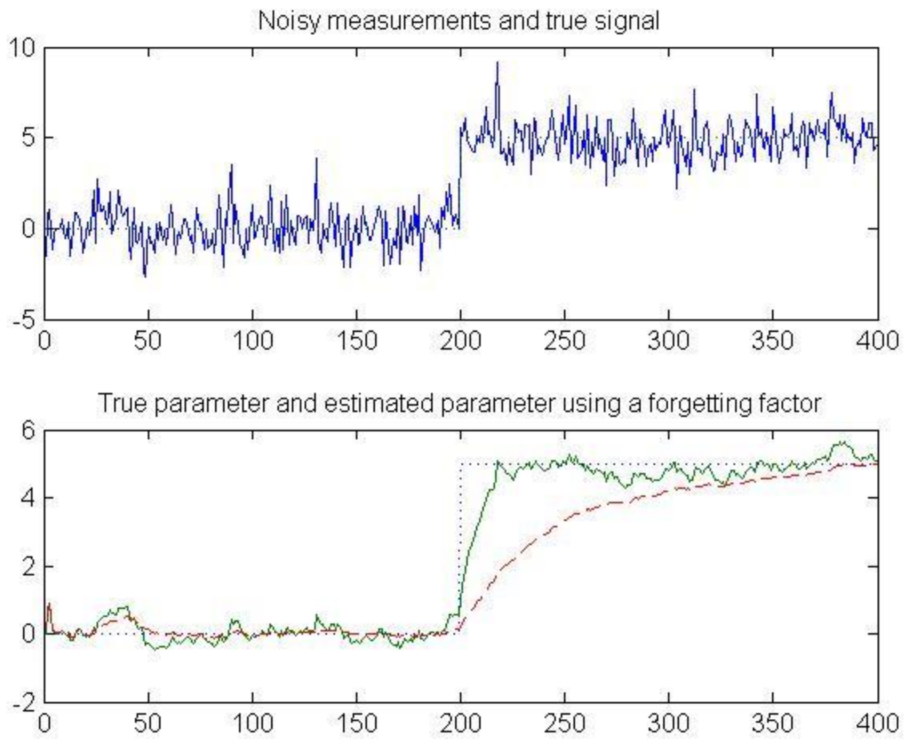

Figure 1. Illustration of RLS with forgetting factor. Upper plot – measured signal y(t). Lower plot- estimated parameter, true parameter (dotted line), RLS estimate with forgetting factor 0.90 (solid line), and with forgetting factor 0.98 (dashed line).

4.2 Recursive identification of a time varying parameters using a Kalman filter.

We will here compare RLS with the Kalman filter approach. We will use data from the following system

$$y(t) = -a(t)y(t-1) + b(t)u(t-1) + e(t)$$

where a(t)=-0.8 for all t and b(t)=1 for t<200 and b(t)=1.5 for t>200.

The result from RLS with a forgetting factor $\lambda = 0.98$ is seen in Figure 2. Note that both the estimated parameters, $\hat{a}(t)\ and\ \hat{b}(t)$, are quite noisy.

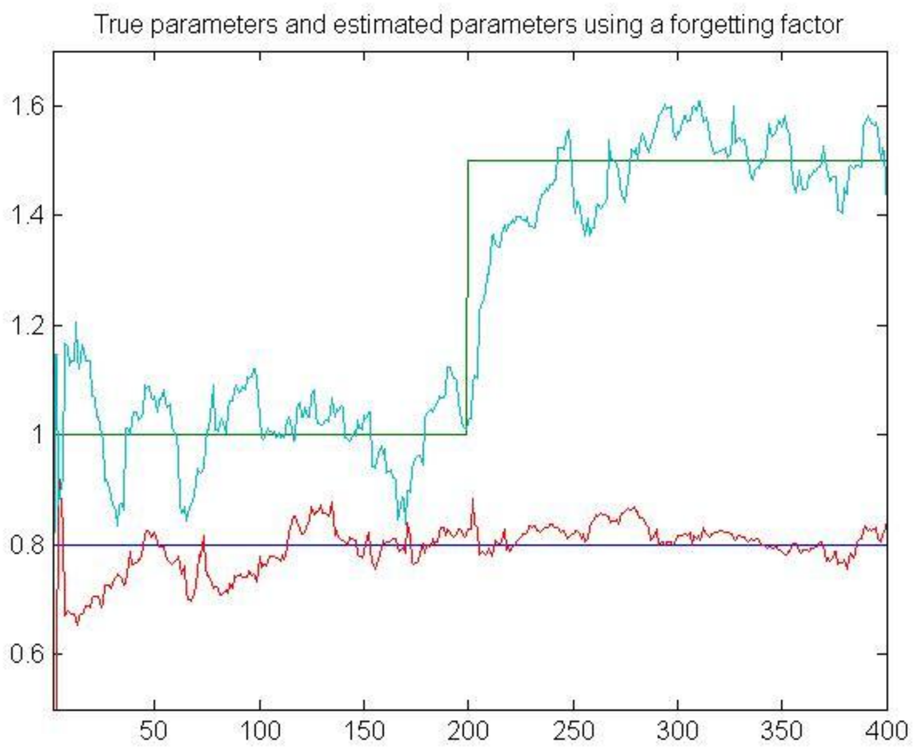

Figure 2. Illustration of RLS with forgetting factor 0.94 (in the figure the a-parameter is shown with opposite sign).

In Figure 3, we show the estimated parameters when a Kalman filter is used. In the design of the Kalman filter we assume that we know that the parameter *a* is not varying. A reasonable choice of R_1 is then

$$R_1 = \begin{bmatrix} 0 & 0 \\ 0 & r \end{bmatrix}$$

where we have set *r=0.01*. As seen from Figure 3, the estimated *a*-parameter is much smoother than for the RLS case (Figure 2). The reason is that the Kalman filter “knows” by the choice of R_1 above that only the parameter *b* may change.

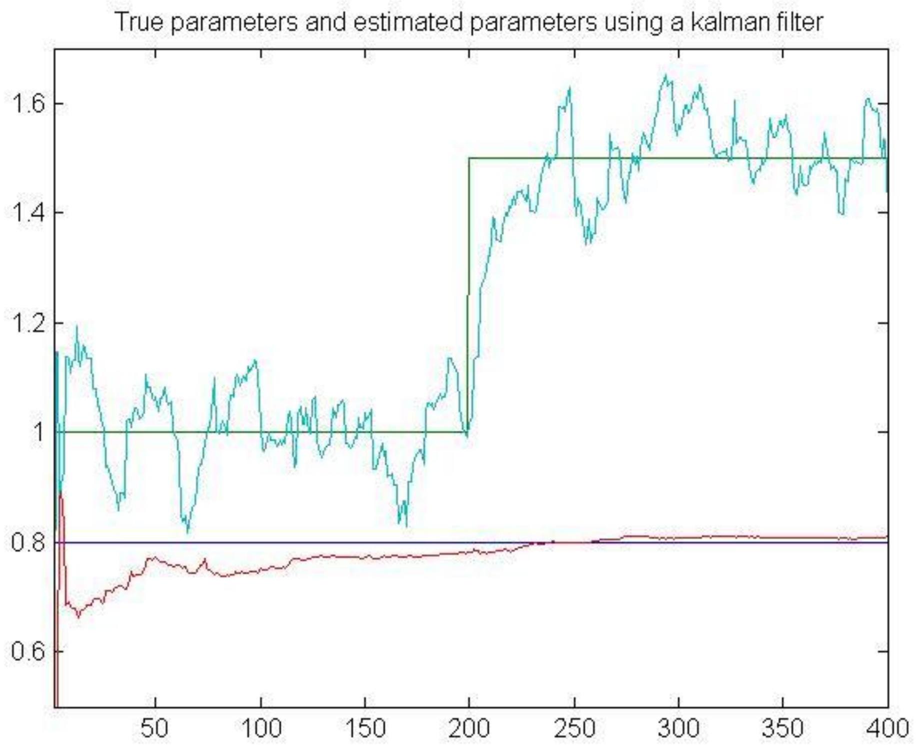

Figure 3. Illustration of the Kalman filter (in the figure the *a*-parameter is shown with opposite sign).

5 Summary

A brief overview of some common recursive identification methods has been given. We have restricted the presentation to linear regression models. The algorithm is called RLS (Recursive Least Squares). Similar ideas can be used to derive recursive algorithms for more general model structures. The problem to track time varying parameters was also treated. The simplest solution is to introduce a forgetting factor (RLS with a forgetting factor). In all recursive algorithms a tradeoff between fast tracking and low noise sensitivity must be made. For cases when it is known that different parameters are likely to change with different rates, a Kalman filter is a suitable choice. Some numerical examples were used to illustrate the algorithms.

Finally we would like to mention that for the case when system has fast parameters that occurs seldom, other methods are suitable to use.

6 Referenser

Söderström, T., & Stoica, P. (1989). *System Identification.* Prentice-Hall International.

Ljung L. and Söderström, T. (1983). Theory and Practice of Recursive *System Identification.* Prentice-Hall International.

Bilaga 1: Härledning av RLS från minstakvadratskattningen

$$\begin{aligned}\hat{\theta}(t) &= \left[\sum_{s=1}^{t} \varphi(s)\varphi^T(s)\right]^{-1} \left(\sum_{s=1}^{t-1} \varphi(s)y(s) + \varphi(t)y(t)\right) \\ &= \left[\sum_{s=1}^{t} \varphi(s)\varphi^T(s)\right]^{-1} \left(\left[\sum_{s=1}^{t-1} \varphi(s)\varphi^T(s)\right]\hat{\theta}(t-1) + \varphi(t)y(t)\right) \\ &= \left[\sum_{s=1}^{t} \varphi(s)\varphi^T(s)\right]^{-1} \left(\left[\sum_{s=1}^{t} \varphi(s)\varphi^T(s) - \varphi(t)\varphi^T(t)\right]\hat{\theta}(t-1) + \varphi(t)y(t)\right) \\ &= \hat{\theta}(t-1) + \left[\sum_{s=1}^{t} \varphi(s)\varphi^T(s)\right]^{-1} \left(\varphi(t)y(t) - \varphi(t)\varphi^T(t)\hat{\theta}(t-1)\right)\end{aligned}$$

Next we set $P(t) = [\sum_{s=1}^{t} \varphi(s)\varphi^T(s)]^{-1}$ which gives

$$\begin{cases}\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)\left(y(t) - \varphi^T(t)\hat{\theta}(t-1)\right) \\ K(t) = P(t)\varphi(t)\end{cases}$$

Note that $P^{-1}(t) = P^{-1}(t-1) + \varphi(t)\varphi^T(t)$. In order to proceed we will use a famous matrix inversion lemma:

$(A + BCD)^{-1} = A^{-1} - A^{-1}B(DA^{-1}B + C^{-1})DA^{-1}$

we get the following recursion for *P(t):*

$$P(t) = P(t-1) - \frac{P(t-1)\varphi(t)\varphi^T(t)P(t-1)}{1 + \varphi^T(t)P(t-1)\varphi(t)}$$

We may also write the updating gain as

$$K(t) = P(t)\varphi(t) = P(t-1)\varphi(t) - \frac{P(t-1)\varphi(t)\varphi^T(t)P(t-1)\varphi(t)}{1 + \varphi^T(t)P(t-1)\varphi(t)} = \frac{P(t-1)\varphi(t)}{1 + \varphi^T(t)P(t-1)\varphi(t)}$$

Bilaga 2: Härledning av beräkningsalgoritm för RLS med glömskefaktor

The criterion J leads to least squares estimate

$$\hat{\theta}(t) = \left[\sum_{s=1}^{t} \lambda^{t-s}\varphi(s)\varphi^T(s)\right]^{-1} \sum_{s=1}^{t} \lambda^{t-s}\varphi(s)y(s)$$

Note that the forgetting factor can be written recursively as $\lambda^{t-s} = \lambda \cdot \lambda^{t-s-1}$.

Next introduce$P(t) = [\sum_{s=1}^{t} \lambda^{t-s}\varphi(s)\varphi^T(s)]^{-1}$. We then have $P^{-1}(t) = \lambda P^{-1}(t-1) + \varphi(t)\varphi^T(t)$. The estimate may then be written recursively as

$$\begin{aligned}\hat{\theta}(t) = P(t)\left(\lambda \sum_{s=1}^{t-1} \lambda^{t-s-1}\varphi(s)y(s) + \varphi(t)y(t)\right) &= P(t)\left(\lambda P^{-1}(t-1)\hat{\theta}(t-1) + \varphi(t)y(t)\right)\\ &= P(t)\left(P^{-1}(t)\hat{\theta}(t-1) + \varphi(t)\left(y(t) - \varphi^T(t)\hat{\theta}(t-1)\right)\right)\\ &= \hat{\theta}(t-1) + P(t)\varphi(t)\left(y(t) - \varphi^T(t)\hat{\theta}(t-1)\right)\end{aligned}$$

Using the matrix inversion lemma on $P^{-1}(t) = \lambda P^{-1}(t-1) + \varphi(t)\varphi^T(t)$ the following recursion for *P(t)* is obtained:

$$P(t) = \frac{1}{\lambda}\left(P(t-1) - \frac{P(t-1)\varphi(t)\varphi^T(t)P(t-1)}{\lambda + \varphi^T(t)P(t-1)\varphi(t)}\right)$$