ECEn 671: Mathematics of Signals and Systems

Randal W. Beard

Brigham Young University

December 13, 2023

Section 1

Batch Least Squares

Least Squares Filtering Problem

Suppose that you have an application, like system identification, where you are trying to estimate a set of parameters from noisy data. For example, suppose that you are trying to estimate the parameters of the discrete-time system

$$\begin{aligned} y[k] = a_1 y[k-1] + a_2 y[k-2] + \cdots + a_n y[k-n] \\ + b_0 u[k] + b_1 u[k-1] + \cdots + b_m u[k-m] \end{aligned}$$

where you know the inputs $u[k]$ and the measure the output $y[k]$ plus noise.

Least Squares Filtering Problem, cont.

Rewrite the measurement at time k as

$$y[k] = \begin{pmatrix} y[k-1] & \cdots & y[k-n] & u[k] & \cdots & u[k-m] \end{pmatrix} \begin{pmatrix} a_1 \\ \vdots \\ a_n \\ b_0 \\ \vdots \\ b_{m-1} \end{pmatrix} + \eta$$

$$= \mathbf{a}_k^\top \mathbf{x} + \eta$$

where η is noise and

$$\mathbf{a}_k^\top = \begin{pmatrix} y[k-1] & y[k-2] & \cdots & y[k-n] & u[k] & \cdots & u[k-m] \end{pmatrix}$$
$$\mathbf{x}^\top = \begin{pmatrix} a_1 & \cdots & a_n & b_0 & \cdots & b_{m-1} \end{pmatrix}.$$

Least Squares Filtering Problem, cont.

Collecting N samples and stacking as a matrix gives

$$\begin{pmatrix} y[1] \\ \vdots \\ y[N] \end{pmatrix} = \begin{pmatrix} \mathbf{a}_1^\top \\ \vdots \\ \mathbf{a}_N^\top \end{pmatrix} \mathbf{x}_N + \begin{pmatrix} \eta_1 \\ \vdots \eta_N \end{pmatrix}$$

$$\implies \mathbf{y}_N = A_N \mathbf{x}_N + \boldsymbol{\eta}_N,$$

where

$$\mathbf{y}_N = \begin{pmatrix} y[1] & \cdots & y[N] \end{pmatrix}^\top$$

$$A_N = \begin{pmatrix} \mathbf{a}_1^\top \\ \vdots \\ \mathbf{a}_N^\top \end{pmatrix}$$

and x_N is the least squares solution given N samples.

Least Squares Filtering Problem, cont.

We know that the batch least squares solution is

$$\mathbf{x}_N^* = \left(A_N^\top A_N\right)^{-1} A_N^\top \mathbf{y}_N$$

While the matrix $A_N^\top A_N$ is always $(n+m) \times (n+m)$, computing $A_N^\top A_N$ requires the storage and multiplication of matrices of the size $N \times (n+m)$ which can become prohibitively large for a large number of samples.

Therefore, computing batch least squares at every sample is not a reasonable strategy.

The recursive least squares (RLS) algorithm solves this problem.

Section 2

Recursive Least Squares Filtering

Recursive Least Squares Filtering

Least squares solution:

$$\mathbf{x}_N^* = \left(A_N^\top A_N\right)^{-1} A_N^\top \mathbf{y}_N$$

Define

$$P_N = \left(A_N^\top A_N\right)^{-1}$$
$$\mathbf{z}_N = A_N^\top \mathbf{y}_N$$

then

$$\mathbf{x}_N^* = \underbrace{P_N}_{(n+m)\times(n+m)} \underbrace{\mathbf{z}_N}_{(n+m)\times 1}$$

where the size of P_N and $\mathbf{z}_N$ are independent of the number of samples N.

Recursive Least Squares Filtering, cont.

Note that after $N-1$ samples

$$P_{N-1}^{-1} \triangleq A_{N-1}^T A_{N-1} = \begin{pmatrix} \mathbf{a}_1 & \cdots & \mathbf{a}_{t-1} \end{pmatrix} \begin{pmatrix} \mathbf{a}_1^T \\ \vdots \\ \mathbf{a}_{t-1}^T \end{pmatrix}$$

$$= \sum_{i=1}^{N-1} \mathbf{a}_i \mathbf{a}_i^T .$$

Receiving a new sample at time N: $y[N] = \mathbf{a}_N^\top \mathbf{x}$, then
Then

$$\begin{aligned} P_N^{-1} &= \sum_{i=1}^{N} \mathbf{a}_i \mathbf{a}_i^T \\ &= \sum_{i=1}^{N-1} \mathbf{a}_i \mathbf{a}_i^T + \mathbf{a}_N \mathbf{a}_N^T \\ &= P_{N-1}^{-1} + \mathbf{a}_N \mathbf{a}_N^T . \end{aligned}$$

Recursive Least Squares Filtering, cont.

Using the matrix inversion lemma:

$$(A + XRY)^{-1} = A^{-1} - A^{-1}X(R^{-1} + YA^{-1}X)^{-1}YA^{-1}$$

gives

$$\begin{aligned} P_N &= \left(P_{N-1}^{-1} + \mathbf{a}_N \mathbf{a}_N^T\right)^{-1} \\ &= P_{N-1} - P_{N-1}\mathbf{a}_N \left(1 + \mathbf{a}_N^\top P_{N-1}\mathbf{a}_N\right)^{-1} \mathbf{a}_N^\top P_{N-1} \\ &= P_{N-1} - \frac{P_{N-1}\mathbf{a}_N \mathbf{a}_N^\top P_{N-1}}{1 + \mathbf{a}_N^\top P_{N-1}\mathbf{a}_N} \end{aligned}$$

where we note that an $(n+m) \times (n+m)$ inverse has been replaced by a 1×1 inverse.

Recursive Least Squares Filtering, cont.

Note that we have found a clever way to **recursively** update $P_N = (A_N^\top A_N)^{-1}$ with new data:

$$P_N = P_{N-1} - \frac{P_{N-1}\mathbf{a}_N\mathbf{a}_N^\top P_{N-1}}{1 + \mathbf{a}_N^\top P_{N-1}\mathbf{a}_N}$$

where $\mathbf{a}_N$ represents the new data.

Recursive Least Squares Filtering, cont.

Similarly

$$
\begin{aligned}
\mathbf{z}_N &= A_N^\top \mathbf{y}_N \\
&= \sum_{i=1}^{N} \mathbf{a}_i y[i] \\
&= \sum_{i=1}^{N-1} \mathbf{a}_i y[i] + \mathbf{a}_N y[N] \\
&= \mathbf{z}_{N-1} + \mathbf{a}_N y[N]
\end{aligned}
$$

Recursive Least Squares Filtering, cont.

Therefore the **exact** least squares solution after N samples is

$$
\begin{aligned}
\mathbf{x}_N &= (A_N^\top A_N)^{-1} A_N^\top \mathbf{y}_N \\
&= P_N \mathbf{z}_N \\
&= \left(P_{N-1} - \frac{P_{N-1}\mathbf{a}_N \mathbf{a}_N^\top P_{N-1}}{1 + \mathbf{a}_N^\top P_{N-1}\mathbf{a}_N} \right) (\mathbf{z}_{N-1} + \mathbf{a}_N y[N]) \\
&= P_{N-1}\mathbf{z}_{N-1} - \frac{P_{N-1}\mathbf{a}_N \mathbf{a}_N^\top P_{N-1}}{1 + \mathbf{a}_N^\top P_{N-1}\mathbf{a}_N} \mathbf{z}_{N-1} \\
&\quad + \left(P_{N-1} - \frac{P_{N-1}\mathbf{a}_N \mathbf{a}_N^\top P_{N-1}}{1 + \mathbf{a}_N^\top P_{N-1}\mathbf{a}_N} \right) \mathbf{a}_N y[N] \\
&= \mathbf{x}_{N-1} - \left(\frac{P_{N-1}\mathbf{a}_N}{1 + \mathbf{a}_N^\top P_{N-1}\mathbf{a}_N} \right) \mathbf{a}_N^\top P_{N-1}\mathbf{z}_{N-1} \\
&\quad + \left(P_{N-1} - \frac{P_{N-1}\mathbf{a}_N \mathbf{a}_N^\top P_{N-1}}{1 + \mathbf{a}_N^\top P_{N-1}\mathbf{a}_N} \right) \mathbf{a}_N y[N]
\end{aligned}
$$

Recursive Least Squares Filtering, cont.

Define (the Kalman gain)

$$\mathbf{k}_N \triangleq \frac{P_{N-1}\mathbf{a}_N}{1+\mathbf{a}_N^\top P_{N-1}\mathbf{a}_N}$$

and note that

$$\begin{aligned}
&\left(P_{N-1} - \frac{P_{N-1}\mathbf{a}_N\mathbf{a}_N^\top P_{N-1}}{1+\mathbf{a}_N^\top P_{N-1}\mathbf{a}_N}\right)\mathbf{a}_N \\
&= \frac{P_{N-1}\mathbf{a}_N(1+\mathbf{a}_N^\top P_{N-1}\mathbf{a}_N) - P_{N-1}\mathbf{a}_N\mathbf{a}_N^\top P_{N-1}\mathbf{a}_N}{1+\mathbf{a}_N^\top P_{N-1}\mathbf{a}_N} \\
&= \frac{P_{N-1}\mathbf{a}_N + P_{N-1}\mathbf{a}_N\mathbf{a}_N^\top P_{N-1}\mathbf{a}_N - P_{N-1}\mathbf{a}_N\mathbf{a}_N^\top P_{N-1}\mathbf{a}_N}{1+\mathbf{a}_N^\top P_{N-1}\mathbf{a}_N} \\
&= \frac{P_{N-1}\mathbf{a}_N}{1+\mathbf{a}_N^\top P_{N-1}\mathbf{a}_N} \\
&= \mathbf{k}_N
\end{aligned}$$

Recursive Least Squares Filtering, cont.

Therefore

$$
\begin{aligned}
\mathbf{x}_N &= \mathbf{x}_{N-1} - \left(\frac{P_{N-1}\mathbf{a}_N}{1 + \mathbf{a}_N^\top P_{N-1}\mathbf{a}_N} \right) \mathbf{a}_N^\top P_{N-1}\mathbf{z}_{N-1} \\
&\quad + \left(P_{N-1} - \frac{P_{N-1}\mathbf{a}_N\mathbf{a}_N^\top P_{N-1}}{1 + \mathbf{a}_N^\top P_{N-1}\mathbf{a}_N} \right) \mathbf{a}_N y[N] \\
&= \mathbf{x}_{N-1} - \mathbf{k}_N\mathbf{a}_N^\top P_{N-1}\mathbf{z}_{N-1} + \mathbf{k}_N y[N] \\
&= \mathbf{x}_{N-1} + \mathbf{k}_N \left(y[N] - \mathbf{a}_N^\top P_{N-1}\mathbf{z}_{N-1} \right) \\
&= \mathbf{x}_{N-1} + \mathbf{k}_N \left(y[N] - \mathbf{a}_N^\top \mathbf{x}_{N-1} \right)
\end{aligned}
$$

Note that $\hat{y}[N] = \mathbf{a}_N^\top \mathbf{x}_{N-1}$ is the predicted output, and $e_N = y[N] - \hat{y}[N]$ is the quantity that is being minimized.

Recursive Least Squares Filtering, interpretation.

$$\underbrace{\mathbf{x}_N}_{\text{new estimate}} = \underbrace{\mathbf{x}_{N-1}}_{\text{old estimate}} + \underbrace{\mathbf{k}_N}_{\text{Kalman gain}} \underbrace{(y[N] - \hat{y}[N])}_{\text{innovation}}$$

where the innovation is the difference between the actual measurement and the predicted measurement.

Summary: Recursive Least Squares Filtering

At time $t = 0$ initialize algorithm with

$$\begin{aligned} P_0 &= \alpha I, \text{ where } \alpha > 0 \text{ is a large number} \\ \mathbf{x}_0 &= 0. \end{aligned}$$

At sample N, collect output $y[N]$ and input $u[N]$ and construct $\mathbf{a}_N$ from using current and past inputs and outputs.

Update the least squares estimate using

$$\begin{aligned} \mathbf{k}_N &= \frac{P_{N-1}\mathbf{a}_N}{1 + \mathbf{a}_t^\top P_{N-1}\mathbf{a}_N} \\ P_N &= P_{N-1} - \mathbf{k}_N \mathbf{a}_N^\top P_{N-1} \\ \mathbf{x}_N &= \mathbf{x}_{N-1} + \mathbf{k}_N(y[N] - \mathbf{a}_N^\top \mathbf{x}_{N-1}). \end{aligned}$$

This is equivalent to a discrete time Kalman filter with stationary dynamics.