EECS 225A Statistical Signal Processing Lecture 25 - 4/28/2020

Lecture 24: CKMS Recursion

Lecturer: Jiantao Jiao *Scribe: Milo Webster and Shafeeq Ibraheem*

The Kalman Filter has a nice property, being that we can allow the state space model parameters (F, G, etc.) to be time variant, and the Kalman Filter formulas look the same. This is quite powerful because we do not require any stationarity assumptions when we apply Kalman filter. This is partly why Kalman Filtering became so popular in its early days.

Previously we also discussed how this particular advantage could be viewed as a disadvantage. In this lecture we discuss how, when we know that our parameters are time invariant, we can speed up computation of the Kalman Filter. This method is called the *Fast Algorithm for Kalman Filtering*, also known as *CKMS Recursion* [KSH00, Chapter 11].

1 Motivation

We first consider the standard state space model, in which we treat the parameters as constant:

$$\begin{cases} X_{i+1} = FX_i + GU_i, \, i \geq 0 \\ Y_i = HX_i + V_i \end{cases} \tag{1}$$

With second order statistics as reflecting constant parameters:

$$\left\langle \begin{bmatrix} U_i \\ V_i \\ X_0 \end{bmatrix}, \begin{bmatrix} U_j \\ V_j \\ X_0 \end{bmatrix} \right\rangle = \begin{bmatrix} Q\delta_{ij} & S\delta_{ij} & 0 \\ S^*\delta_{ij} & R\delta_{ij} & 0 \\ 0 & 0 & \Pi_0 \end{bmatrix} \tag{2}$$

In the past we haven't emphasized the dimension of the variables. Here, we assume the following dimensions:

$$X_i \in \mathbb{C}^{n \times 1} \tag{3}$$

$$Y_i \in \mathbb{C}^{p \times 1} \tag{4}$$

$$U_i \in \mathbb{C}^{m \times 1} \tag{5}$$

And in general we have the following properties:

$$p \leq n \tag{6}$$

$$m \leq n \tag{7}$$

The first property ($p \leq n$) is the case because our state summarizes everything that we know. And we can view Y as a reduced observation: a partial observation of the state. It follows from this that it would be ridiculous to assume that the dimension of this observation is larger than that of our state, considering it captures less information.

The second property ($m \leq n$) is the case because m is the dimension of our input, and this input together with the parameter G has an equivalent dimension to our state. And so having the dimension of the input be larger than our state doesn't make sense given that the same (in terms of what the state sees) could be represented in a lower dimension, given that the state only sees the dimension of G's rows.

2 Computational Complexity of the Kalman Filter

We first look at the computational complexity of the regular Kalman Filter, for a baseline to compare our "fast" version to. For the predictor version of the Kalman Filter, we have the following system:

$$e_i = Y_i - H\hat{X}_i \tag{8}$$

$$\hat{X}_{i+1} = F\hat{X}_i + K_i R_{e,i}^{-1} e_i,\ \hat{X}_0 = 0 \tag{9}$$

Note that here we use the term $K_i R_{e,i}^{-1}$ in place of $K_{p,i}$ because it will later facilitate our derivation of the "fast" recursion. Please do not confuse K_i with either the predicted/filtered Kalman gain. We can compute these two new terms according to the following formulas:

$$K_i = FP_iH^* + GS \tag{10}$$

$$R_{e,i} = R + HP_iH^* \tag{11}$$

$$P_{i+1} = FP_iF^* + GQG^* - K_iR_{e,i}^{-1}K_i^*a,\ P_0 = \Pi_0, \tag{12}$$

which was derived when we introduced the predicted version Kalman filter.

Now, in general, the biggest computational complexity is in computing the P_{i+1}. For example, the first term (FP_iF^*) consists of 3 matrix multiples, each having dimension $n \times n$. With the naive/brute force computation, the complexity is $O(n^3)$. This computation, in general, is the bottle neck for naive/brute force computation.

3 Reducing the Computational Complexity

The key idea, here, is that *propogating P_i is bad*. We should define recursions for δP_i, instead of for P_i directly, which we define as follows:

$$\delta P_i \triangleq P_{i+1} - P_i,\ i \geq 0 \tag{13}$$

Previously we used the Ricatti equation to go from P_i to P_{i+1}, but now we need to know how to go from δP_i to δP_{i+1}. This is addressed in the following theorem.

Recall that we have defined

$$F_{p,i} \triangleq F - K_iR_{e,i}^{-1}H. \tag{14}$$

Theorem 1. *Generalized Stokes Identity]*

$$\delta P_{i+1} = F_{p,i}[\delta P_i - \delta P_iH^*R_{e,i+1}^{-1}H\delta P_i]F_{p,i}^* \tag{15}$$

This theorem immediately implies that

$$rank(\delta P_{i+1}) \leq rank(\delta P_i) \tag{16}$$

This observation, that the rank of δP_i will not increase with i, is possibly the most significant observation in the derivation of the "fast" algorithm. And now, if we can show that the rank of δP_0 is low rank, then we have an opportunity to eliminate unnecessary computation in this recursion via low rank matrix computation.

Now to derive δP_0, we have the following (where $P_0 = \Pi_0$ and P_1 is computed using the Ricatti equations):

$$\delta P_0 = P_1 - P_0 \tag{17}$$

$$\delta P_0 = F\Pi_0F^* + GQG^* - K_0R_{e,0}^{-1}K_0^* - \Pi_0 \tag{18}$$

Our goal is to claim that this expression is low rank, which we can do by breaking it into a number of cases. Note that in all of these cases (and the entire lecture), we assume that $R > 0$.

Case 1: $\Pi_0 = 0$

$$\delta P_0 = 0 + GQG^* - K_0 R_{e,0}^{-1} K_0^* + 0 \tag{19}$$

$$\delta P_0 = G(Q - SR^{-1}S^*)G^* \tag{20}$$

The matrix G, here, first appeared from our state equation $X_{i+1} = FX_i + GU_i$. From this we know that G has dimension $n \times m$. And therefore, we have that:

$$rank(\delta P_0) \leq m \tag{21}$$

Which implies that the rank of δP_0 in this case is small.

Case 2: Stationary Process

By stationary process we mean that we choose the initial variance Π_0 such that our state X_i is a stationary process. We can do this by ensuring that Π_0 satisfies the Lyapunov Equation,

$$\Pi_0 = F\Pi_0 F^* + GQG^* \tag{22}$$

which has a PSD solution if all the eigenvalues of F lie in the unit disk $|\lambda_i| < 1$. It's important to note here that *initializing at* Π_0 *does not make a Kalman Filter a time-invariant filter.* Our KF still takes time to converge to a particular (time-invariant) filter. Now, in this case we can also easily compute δP_0:

$$\delta P_0 = -K_0 R_{e,0}^{-1} K_0^* \tag{23}$$

Now note that $K_0 = F\Pi_0 H^* + GS$, where F has dimension $n \times n$ and the dimension of H^* is $n \times p$, which gives that K_0 has dimension $n \times p$. This then tells us that,

$$rank(\delta P_0) \leq p \tag{24}$$

Now, we can apply this knowledge to the following theorem:

Theorem 2. *Propogation of δP_i] If we factorize δP_0,*

$$\delta P_0 = L_0 M_0 L_0^* \tag{25}$$

Where, $rank(\delta P_0) = \alpha$. And we require the matrix L_0 to meet,

$$L_0 \in \mathbb{C}^{n \times \alpha} \tag{26}$$

Then we want this M_0 to be Hermitian, meeting

$$M_0 \in \mathbb{C}^{\alpha \times \alpha} \tag{27}$$

Then we have that,

$$L_{i+1} = (F - K_i R_{e,i}^{-1} H)L_i = F_{p,i} L_i \tag{28}$$

$$M_{i+1} = M_i - M_i L_i^* H^* R_{e,i+1}^{-1} H L_i M_i,\ i \leq 0 \tag{29}$$

Proving this theorem is actually quite simple because we can use the Stoked Identity. To understand the above equation we can first look at δP_0. Given rank α of this term, we can actually factorize it into this form with computational complexity $O(n^2\alpha)$, where previously (in the general case) we need $O(n^3)$. Now, for the computational complexity of L_{i+1}, we see that $F_{p,i}$ has dimension $n \times n$, while L_i has dimension $n \times \alpha$. This results in a complexity of $O(n^2\alpha)$, just like for δP_0. Finally, we address the complexity of M_{i+1}, but we actually don't really care what this term's complexity is because in reality it is usually quite small. Even though it has complexity $O(\max(\alpha^3, \alpha np, np^2))$, where α is generally small. Now, after this propagation of δP_i, we address a related topic to round out our understanding.

Theorem 3. *CKMS Recursion*

$$\begin{cases} K_{i+1} = K_i - FL_i R_{r,i}^{-1} L_i^* H^* \\ L_{i+1} = FL_i - K_i R_{e,i}^{-1} HL_i \\ R_{e,i+1} = R_{e,i} - HL_i R_{r,i}^{-1} L_i^* H^* \\ R_{r,i+1} = R_{r,i} - L_i^* H^* R_{e,i}^{-1} HL_i \end{cases} \tag{30}$$

Conditioned on,

$$R_{r,i} \triangleq -M_i^{-1} \tag{31}$$

$$P_{i+1} = P_0 + \sum_{j=0}^{i} \delta P_j = \Pi_0 - \sum_{j=0}^{i} L_j R_{r,j}^{-1} L_j^* \tag{32}$$

Initialized with,

$$\Pi_0 \to \begin{cases} K = F\Pi_0 H^* + GS \\ R_{e,0} = R + H\Pi_0 H^* \\ \delta P_0 = P_1 - P_0 \end{cases} \tag{33}$$

4 Array Algorithm for CKMS Recursion

We deal with a special case, being that $\Pi_0 = 0$. Now, we know that,

$$\delta P_0 = P_1 - P_0 = G(Q - SR^{-1}S^*)G^* \tag{34}$$

$$\delta P_0 = GQ^S G^* = \bar{L}_0 \bar{L}_0^* \tag{35}$$

Where we have:

$$\bar{L}_0 \triangleq G(Q^S)^{\frac{1}{2}} \tag{36}$$

Note that we have dimensions here for $\bar{L}_0$ of $n \times m$, where $\alpha \leq m$. And we want to speed up computation of:

$$\delta P_i = P_{i+1} - P_i = \bar{L}_i \bar{L}_i^* \tag{37}$$

We can do this with the following methods:

$$\begin{pmatrix} R_{e,i}^{\frac{1}{2}} & H\bar{L}_i \\ \bar{K}_{p,i} & F\bar{L}_i \end{pmatrix} H = \begin{pmatrix} X & 0 \\ Y & Z \end{pmatrix} \tag{38}$$

Where H is an orthogonal unitary matrix and $\bar{K}_{p,i} \triangleq K_{p,i} R_{e,i}^{-\frac{*}{2}}$. We can verify this result by showing that, given equivalence with $A = B$, $AA^* = BB^*$. This gives the following:

$$\begin{cases} X = R_{e,i+1}^{\frac{1}{2}} \\ Y = \bar{K}_{p,i+1} = K_{p,i+1} R_{e,i+1}^{-\frac{*}{2}} \\ Z = \bar{L}_{i+1} \end{cases} \tag{39}$$

Now we need to show that the dimension of the matrix for this array algorithm is much smaller, in order for it to achieve a speedup. Observe that $R_{e,i}$ measures the variance of the innovations, having dimension p. $\bar{K}_{p,i}$ has n rows, because $K_{p,i}$ takes the observation Y to have the state dimension n. So the full $\bar{K}_{p,i}$ matrix is $n \times p$. Then, the dimension of H is $p \times n$, while the dimension of $\bar{L}_i$ is $n \times \alpha$. And so $H\bar{L}_i$ has dimension $p \times \alpha$. Given these we can observe that our block matrix that is multiplied with the unitary matrix H has dimension $(p+n) \times (p+m)$. The complexity of the array algorithm can be found in [KSH00, Page 54].

References

[KSH00] Thomas Kailath, Ali H Sayed, and Babak Hassibi. *Linear estimation*. Prentice Hall, 2000.