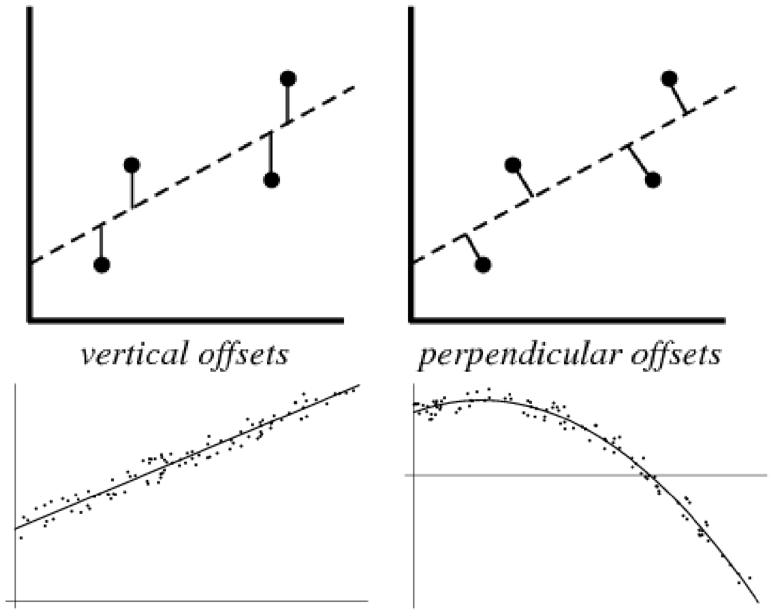
2018.10.22 Lecture Notes – Group 6

이태헌 (20185230), 손선민 (20184332), 신훈 (20183326), Haritz Puerto San Roman (20184669)

Introduction to Least Squares Estimation (Previous class)

In the previous class, we learned that LSE is a standard approach in regression analysis to find the best-fitting curve to a given set of points by minimizing the sum of the square of the offsets of the points from the curve.



In addition, we also studied the recursive least squares algorithm, a recursive algorithm to obtain that best-fitting curve.

Famous algorithms to compute LSE are LASSO and Ridge regression. Both are regularized least squares methods, so they solve the LS problem using regularization to constrain the solution.

Recursive Least Squares Estimation (today's class)

The cost function is as follows:

$$\mathrm{J}_{\mathrm{t}}(\theta) = \sum_{i=1}^{t} \alpha^{t-i}\, e^{2}(i)$$

Analyzing the formula we can observed that when i increases, α, the forgetting factor, gets smaller. This implies that the weight assigned to the error decreases with time, in other words, recent errors have a bigger weight that old errors. If we use a large α, the decrease of the weight will be slow while a small α will imply a fast decrease.

A possible problem that may appear in this algorithm is that the system can become steady stable which means that $\mathrm{P_t}$ diverges, it does not converge to a value, so $\mathrm{P_t} > P_{t-1}$ always. A possible solution is to reset P to a certain value. Typically, after 10 or 20 iterations P is reset this way:

$$\mathrm{P_{t*}} = \mathrm{kI}$$

where I is the identity matrix and k a number greater than zero.

Orthogonal Projection

Let U_m, spanning the whole m-dim space, be

$$U_m = [\,\bar{u}(1), \bar{u}(2), \dots, \bar{u}(m)\,]$$

In orthogonal projection, each datum represents a dimension. As the cardinality of the data increases, the parameters also increase. This method finds the least square solution in exactly m (the dimension of θ) steps. However, this method is unstable and sensitive to errors since the impact of former data does not decrease.

The conjugate gradient method is an algorithm to compute the numerical solution of particular systems of linear equations, namely those whose matrix is symmetric and positive-definite.

Gram–Schmidt process is a method for orthonormalizing a set of vectors in an inner product space, most commonly the Euclidean space, Rn, equipped with the standard inner product.

We define the projection operator by

$proj_u(v) = \frac{<u,v>}{<u,u>} u,$

Where $< \mathrm{u}, \mathrm{v} >$ denotes the inner product of the vectors u and v. This operator projects the vector v orthogonally onto the line spanned by vector u.

The Gram-Schmidt process works as follow:

$u_1 = v_1,$ $\qquad e_1 = \frac{u_1}{\|u_1\|}$

$u_2 = v_2 - proj_{u_1}(v_2),$ $\qquad e_2 = \frac{u_2}{\|u_2\|}$

$u_3 = v_3 - proj_{u_1}(v_3) - proj_{u_2}(v_3),$ $\qquad e_3 = \frac{u_3}{\|u_3\|}$

$u_4 = v_4 - proj_{u_1}(v_4) - proj_{u_2}(v_4) - proj_{u_3}(v_4),$ $\qquad e_4 = \frac{u_4}{\|u_4\|}$

…

$$u_k = v_k - \sum_{j=1}^{k-1} proj_{u_j}(v_k), \quad e_k = \frac{u_k}{\|u_k\|}$$

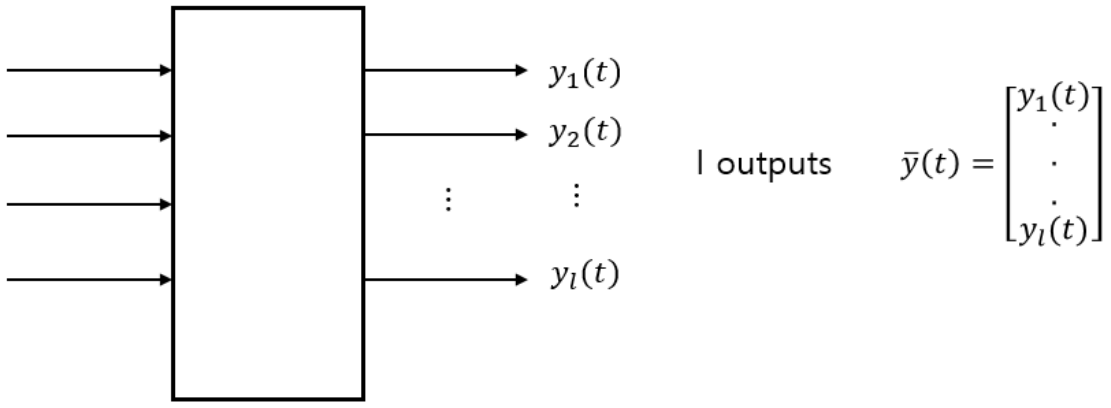
The sequence u_1, …, u_k is the required system of orthogonal vectors, and the normalized vectors e_1, …, e_k form an orthonormal set. The calculation of the sequence u_1, …, u_k is known as Gram-Schmidt orthogonalization, while the calculation of the sequence e_1, …, e_k is known as Gram-Schmidt orthonormalization as the vectors are normalized.

PCA is a statistical procedure that uses an orthogonal transformation to convert a set of observations possibly correlated into a set of linearly uncorrelated variables called principal components. PCA and least squares use a similar error metric, but while PCA treats all dimensions equally LLS treats some dimensions with more preference than others.

Multi-Output Weighted least square estimation

It is similar to single input least square estimation but the dimension of the output is different. The below figure is a diagram of the multi-output system. It has i number of outputs

$$\hat{y}_i(t) = \bar{u}_i^T(t)\theta$$



$y_1(t)$
$y_2(t)$
$y_l(t)$
l outputs
$\bar{y}(t) = \begin{bmatrix} y_1(t) \\ \cdot \\ \cdot \\ \cdot \\ y_l(t) \end{bmatrix}$

The outputs can be changed into matrix form:

$$\hat{\bar{y}}(t) = \begin{bmatrix} \bar{u}_1^T \\ \vdots \\ \bar{u}_l^T \end{bmatrix} \bar{\theta} = \bar{U}^T(t)\bar{\theta}$$

The error vector is defined like this:

$$\bar{e}(t) = \begin{bmatrix} e_1 \\ \vdots \\ e_l \end{bmatrix} = \bar{y}(t) - U^T(t)\bar{\theta} \ ,$$

and the cost function is expressed as a weighted sum of squared each error terms.

$$J_t(\vartheta) = \sum_{i=1}^{t} U^T(i)W\bar{y}(i)$$

If we find $\hat{\vartheta}$ that minimizes the cost function, we would have found the weighted least square solution. It can be easily obtained if we follow same way in the case of single output.

The form of $\hat{\theta}$ is summarized as follows.

$$\hat{\theta}(t) = P_t B_t$$

where $P_t = [\sum_{i=1}^{t} U^T(i)WU(i)]^{-1}$ and $B_t = \sum_{i=1}^{t} U^T(i)W\bar{y}(i)$

Multi-Output Weighted recursive least square estimation

It is also similar to the single output recursive least square estimation but calculations are a bit more complicated because of the matrix operations.

The expansion of the equation is as follows

$$P_t^{-1} = \sum_{i=1}^{t} U^T(i)WU(i) = P_{t-1}^{-1} + U^T(t)WU(t)$$

$$P_{t-1} = P_t + P_t U^T(t)WU(t)P_{t-1}$$

$$P_t U^T(t) = P_{t-1}U^T(t)W\{W^{-1} + U(t)P_{t-1}U^T(t)\}^{-1}$$

$$P_{t-1} - P_t = P_{t-1}U^T(t)\{W^{-1} + U(t)P_{t-1}U^T(t)\}^{-1}U(t)P_{t-1}$$

$$P_t = P_{t-1} - P_{t-1}U^T(t)\{W^{-1} + U(t)P_{t-1}U^T(t)\}^{-1}U(t)P_{t-1}$$

The recursive least square form is as follows

$$\hat{\bar{\theta}}(t) - \hat{\bar{\theta}}(t-1) = K_t\left(\bar{y}(t) - U^T(t)\hat{\bar{\vartheta}}(t-1)\right) = P_t B_t - P_{t-1}B_{t-1}$$

$$= \{P_{t-1} - P_{t-1}U^T(t)\{W^{-1} + U(t)P_{t-1}U^T(t)\}^{-1}U(t)P_{t-1}\}\{B_{t-1} + U^T(t)W\bar{y}(t)\} - P_{t-1}B_{t-1}$$

$$= P_{t-1}U^T(t)\{W^{-1} + U(t)P_{t-1}U^T(t)\}^{-1}\{\bar{y}(t) - U(t)P_{t-1}B_{t-1}\}$$

$$= P_{t-1}U^T(t)\{W^{-1} + U(t)P_{t-1}U^T(t)\}^{-1}\left\{\bar{y}(t) - U(t)\hat{\bar{\vartheta}}(t-1)\right\}$$

Comparing the above equation,

$$K_t = P_{t-1}U^T(t)\{W^{-1} + U(t)P_{t-1}U^T(t)\}^{-1}$$

Therefore, the weighted recursive least square method of multi output system is summarized as follows.

$$\hat{\bar{\theta}} = \hat{\bar{\theta}}(t-1) + P_{t-1}U^T(t)\{W^{-1} + U(t)P_{t-1}U^T(t)\}^{-1}\left(\bar{y}(t) - U^T(t)\hat{\bar{\vartheta}}(t-1)\right)$$

$$P_t = P_{t-1} - P_{t-1}U^T(t)\{W^{-1} + U(t)P_{t-1}U^T(t)\}^{-1}U(t)P_{t-1}$$

Kalman Filter

Kalman filter is an optimal state estimator that takes into account two kinds of noise: sensor and system noise. The sensor and the system may have noise, so the orders given to the system may not be executed correctly. For example, the robot may move forward 49 meters instead of 50. It is usually assumed that the noise follows a Gaussian distribution. In addition, Kalman filters are real time state estimators.

The model and sensor equations are summarized as follows:

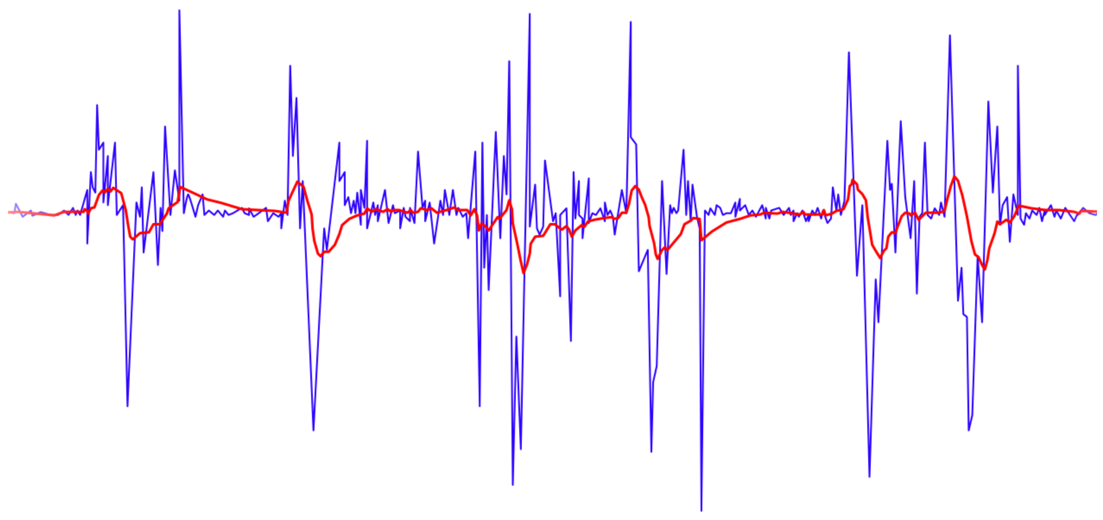
1. $x_{t+1} = A_t x_t + B_t u_t + G_t w_t$, where

$x_{t+1} =$ the assumed true state at time $t + 1$
$A_t =$ the state-transition model
$B_t =$ the control-input model applied to control vector u_t
$w_t =$ process noise $\epsilon\ \mathbb{R}^{n\times 1}$

2. $y_t = H_t x_t + v_t$, where

$y_t =$ measurement of true state x_t
$v_t =$ sensor (observation) noise $\epsilon\ \mathbb{R}^{l\times 1}$
$H_t =$ the observation model

Below is a diagram that shows data with noise (blue) and data processed through the Kalman filter (red). It can be observed that the data resulting from the Kalman filter is much less "jumpy".



After further defining the following:

R_t = covariance of the observation noise
Q_t = covariance of the process noise
$\mathrm{E}[\xi]$ = the expected value of ξ
$C_v(\mathrm{t}, \mathrm{s})$ = Covariance function of observation noise from t to s
$C_w(\mathrm{t}, \mathrm{s})$ = Covariance function of process noise from t to s

we can say that

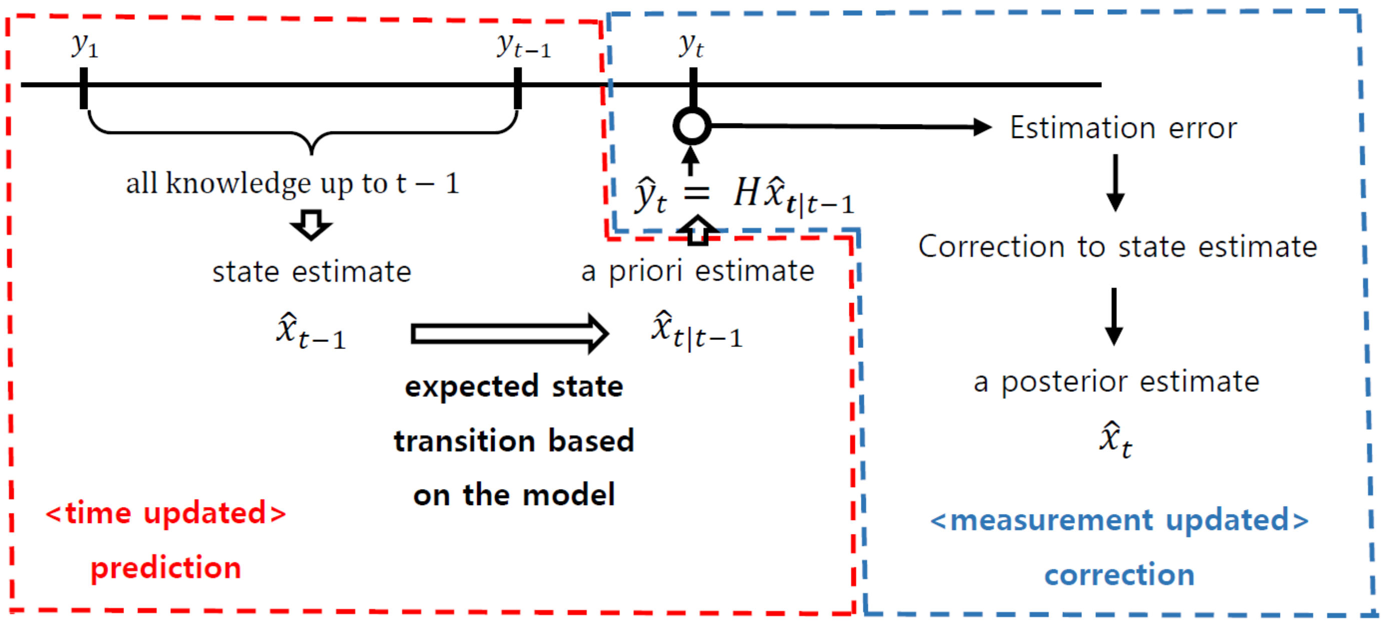
$$C_v(\mathrm{t}, \mathrm{s}) = \mathrm{E}[v_t v_s^T] \epsilon\ \mathbb{R}^{l \times l}$$
$$C_w(\mathrm{t}, \mathrm{s}) = \mathrm{E}[w_t w_s^T] \epsilon\ \mathbb{R}^{n \times n}$$

Also, assuming $\mathrm{E}[w_t] = 0, \mathrm{E}[v_t] = 0$ (zero noise),

1. $C_V(\mathrm{t}, \mathrm{s}) = \mathrm{E}[v_t v_s^T] = 0 \quad \forall t \neq s$
   $C_V(t, s) = R_t \qquad t = s$
2. $C_W(\mathrm{t}, \mathrm{s}) = 0 \qquad \forall t \neq s$
   $C_W(\mathrm{t}, \mathrm{s}) = Q_t \qquad t = s$
3. $C_{VW}(w_t, v_s^T) = 0$

- Prediction and Correction

Kalman filter can be used to minimize the state error. The filter gain is used to minimize the cost function. It has 2 steps. The first step is the prediction part or a priori estimate. It estimates the state using only model information. The second step is the correction part or a posteriori estimate. It uses sensor information to correct the state estimation. The below figure shows 2 steps of optimal state estimation of Kalman filter.



$J = E[(\hat{x}_t - x_t)^T(\hat{x}_t - x_t)]$

From the diagram, we define as follows:

$\hat{x}_{t|t-1}$ = the a priori (predicted) state estimate at t

$\tilde{z}_t$ = measurement pre-fit residual

$\hat{x}_{t|t}$ = the a posteriori state (expected) estimate at t

Then, we can say that

$$\hat{x}_{t|t-1} = A_t\hat{x}_{t-1|t-1} + B_t u_t$$
$$\tilde{z}_t = y_t - H\hat{x}_{t|t-1}$$
$$\hat{y}_t = H\hat{x}_{t|t-1}$$
$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t\hat{y}_t$$
$$\hat{x}_t = \hat{x}_{t|t-1} + K_t(y_t - H\hat{x}_{t|t-1})$$

References

1. https://en.wikipedia.org/wiki/Least_squares
2. https://en.wikipedia.org/wiki/Conjugate_gradient_method
3. https://en.wikipedia.org/wiki/Gram%E2%80%93Schmidt_process
4. https://en.wikipedia.org/wiki/Principal_component_analysis
5. https://blog.maddevs.io/reduce-gps-data-error-on-android-with-kalman-filter-and-accelerometer-43594faed19c
6. https://en.wikipedia.org/wiki/Kalman_filter