# CS572 Lecture Note: October 29

Lecture date: 2018-10-29

Due date: 2018-11-05

Team 5:

20186414 Gautier Clerc, 20184377 Namhyung Lee, 20174337 Hongmin Yang, 20174335 Juneyong Yang

# 0. Least Squares

## 1. Least Squares Estimation of a Linear Model

Input sequence(n points with m dimensions): $A \in \mathcal{R}^{n \times m}$

Output sequence(n corresponding observations): $y \in \mathcal{R}^n$

Model parameters: $x \in \mathcal{R}^m$

Find the x that minimizes the following residual sum of squares:

$$\min_x \|Ax - y\|_2^2$$

$$\begin{aligned} RSS &= \|Ax - y\|_2^2 \\ &= (Ax - y)^T (Ax - y) \\ &= x^T A^T Ax - x^T A^T y - y^T Ax + y^T y \end{aligned}$$

The gradient must be zero:

$$\begin{aligned} \nabla_x RSS &= x^T A^T Ax - x^T A^T y - y^T Ax + y^T y \\ &= -2A^T y + 2A^T Ay = 0 \end{aligned}$$

The analytical solution: $x = (A^T A)^{-1} A^T y$

## 2. Analysis

The matrix $A^T A$ Acts as a covariance matrix between the dimensions of x.

$$A^T A = \begin{bmatrix} x[1,1] & x[1,2] & \cdots & x[1,n] \\ x[2,1] & x[2,2] & \cdots & x[2,n] \\ \vdots & \vdots & \ddots & \vdots \\ x[m,1] & x[m,2] & \cdots & x[m,n] \end{bmatrix} \begin{bmatrix} x[1,1] & x[2,1] & \cdots & x[m,1] \\ x[1,2] & x[2,2] & \cdots & x[m,2] \\ \vdots & \vdots & \ddots & \vdots \\ x[1,n] & x[2,n] & \cdots & x[m,n] \end{bmatrix}$$

$$= \begin{bmatrix} (x[1,1]^2 + x[1,2]^2 + \cdots + x[1,n]^2) \to Var(\text{dim } 1) & \cdots \\ (x[2,1]x[1,1] + x[2,2]x[1,2] + \cdots + x[2,n]x[1,n]) \to Cov(\text{dim } 2, \text{dim } 1) & \cdots \\ \vdots & \vdots \ddots \vdots \\ & \cdots \end{bmatrix}$$

If the system is an autoregressive process, then the matrix $A^T A$ approximates the autocorrelation matrix.

# 1. Recursive Least Squares(with the forgetting factor)

## 1. Motivation

Least squares estimation requires matrix inversion which is very expensive:

$\theta = (A^T A)^{-1} A^T y$ ---> must invert $A^T A$

Doing this every time we get a new data point is very inefficient.

$\theta_t = (A_t^T A_t)^{-1} A_t^T y_t$ ---> invert $A_t^T A_t$

$\theta_{t+1} = (A_{t+1}^T A_{t+1})^{-1} A_{t+1}^T y_{t+1}$ ---> need to invert $A_{t+1}^T A_{t+1}$ again from scratch

Can we get the new parameter estimate without doing it over from scratch?

In other words, we want to obtain $\theta_{t+1}$ easily from $\theta_t$, without a fresh matrix inversion.

## 2. Problem Formulation

Desired output sequence up to time t(t corresponding observations): $\mathbf{y} \in \mathcal{R}^t$

Desired output at time t: $y[t] \in \mathcal{R}$

Estimated parameters at time step t: $\theta_t \in \mathcal{R}^m$

Input sequence at time t: $u_t = [\, x_1 \quad x_2 \quad \cdots \quad x_m \,]_t^T \in \mathcal{R}^m$

Noise at time t: $e[t] \in \mathcal{R}$

Our linear model: $y[t] = u_t \theta_t + e[t]$

Given $u$'s and $\mathbf{y}$, find the $\theta$ that minimizes the error(noise).

We want to solve the following: $\min \sum_{i=0}^{t-1} \lambda^{(t-1)-i} \|error[i]\|^2$

Where $\lambda \in (0 < \lambda < 1)$ is the forgetting factor, making errors further in the past less important.

## 3. The Matrix Inversion Lemma

Let $A \in \mathcal{R}^{M \times M}$, $B \in \mathcal{R}^{M \times M}$, $D \in \mathcal{R}^{M \times M}$ positive definite. Also, $C \in \mathcal{R}^{M \times N}$. if $A = B^{-1} + CD^{-1}C^T$, Then $A^{-1} = B - BC(D + C^T BC)^{-1} C^T B$.

## 4. Derivation

At time t-1: $\Phi_{t-1} = \sum_{n=0}^{t-1} \lambda^{(t-1)-i} u_i u_i^T$

$z_{t-1} = \sum_{t=0}^{t-1} \lambda^{(t-1)-i} u_i y[i]$

$\Phi_{t-1} \theta_{t-1} = z_{t-1}$

$\theta_{t-1} = \Phi_{t-1}^{-1} z_{t-1}$

The $\Phi$ works like $(A^T A)^{-1}$ and $z$ works like $A^T y$.

Current time:

$\Phi_t = \sum_{i=0}^{t} \lambda^{t-i} u_i u_i^T = \sum_{i=0}^{t-1} \lambda^{t-i} u_i u_i^T + u_t u_t^T = \lambda(\sum_{i=0}^{t-1} \lambda^{(t-1)-i} u_i u_i^T) + u_t u_t^T$

$\Phi_t = \lambda \Phi_{t-1} + u_t 1 u_t^T$

$\Phi_t^{-1} = \lambda^{-1}\Phi_{t-1}^{-1} - \frac{\lambda^{-1}\Phi_{t-1}^{-1} u_t u_t^T \lambda^{-1}\Phi_{t-1}^{-1}}{1+u_t^T\lambda^{-1}\Phi_{t-1}^{-1}u_t}$

$P_t = \Phi_t^{-1} = \lambda^{-1}P_{t-1} - \frac{\lambda^{-2}P_{t-1}u_t u_t^T P_{t-1}}{1+u_t^T\lambda^{-1}P_{t-1}u_t} = \lambda^{-1}P_{t-1} - \lambda^{-1}K_t u_t^T P_{t-1}$

where the gain K is given as:

$K_t = \frac{\lambda^{-1}P_{t-1}u_t}{1+u_t^T\lambda^{-1}P_{t-1}u_t} = (\lambda^{-1}P_{t-1} - \lambda^{-1}K_t u_t^T P_{t-1})u_t = P_t u_t$

$$
\begin{aligned}
z_t &= \sum_{i=0}^{t} \lambda^{t-i} u_i y[i] \\
&= \sum_{i=0}^{t-1} \lambda^{t-i} u_i y[i] + u_t y[t] \\
&= \lambda(\sum_{i=0}^{t-1} \lambda^{(t-1)-i} u_i y[i]) + u_t y[t] \\
&= \lambda z_{t-1} + u_t y[t]
\end{aligned}
$$

The parameter:

$$
\begin{aligned}
\theta_t &= \Phi_t^{-1} z_t = P_t z_t \\
&= P_t(\lambda z_{t-1} + u_t y[t]) \\
&= \lambda P_t z_{t-1} + P_t u_t y[t] \\
&= P_{t-1} z_{t-1} - K_t u_t^T P_{t-1} z_{t-1} + P_t u_t y[t] \\
&= \theta_{t-1} - K_t u_t^T \theta_{t-1} + K_t y[t] \\
&= \theta_{t-1} - K_t(u_t^T \theta_{t-1} - y[t])
\end{aligned}
$$

To get the new parameter estimate, modify previous parameter estimate according to the error multiplied by the gain.

## 5. The Algorithm

Repeat:

$\theta_t = \theta_{t-1} + \frac{P_{t-1}u_t}{\lambda+u_t^T P_{t-1}u_t}(y[t] - \theta_{t-1}^T u_t)$

$P_t = \lambda^{-1}(P_{t-1} - \frac{P_{t-1}u_t u_t^T P_{t-1}}{\lambda+u_t^T P_{t-1}u_t})$

## 6. Analysis: The Forgetting Factor

The forgetting factor determines the "memory" capacity.

Small $\alpha$: errors in the past are rapidly discarded, good for rapidly changing processes

Large $\alpha$: errors in the past linger for a long time, good for slowly changing processes

Drawback: when the system is in a stationary state, the $P_t$ matrix might blow up exponentially: $P_t \simeq \frac{1}{\alpha}P_{t-1}$

We remedy this blowup with covariance resetting, in which we reset $P_t$ with $P_t = kI$, $0 < k < \infty$ after typically 10~20 iterations.

# 2. Orthogonal Projections

what if the forgetting factor is 0? ---> orthogonal projections

This method finds the least square solution in exactly m(dimension of parameters) steps.

Gram-Schmidt-like iterative procedure:

$\theta_t = \theta_{t-1} + \frac{P_{t-1}u_t}{u_t^T P_{t-1} u_t}(y[t] - \theta_{t-1}^T u_t)$

$P_t = P_{t-1} - \frac{P_{t-1}u_t u_t^T P_{t-1}}{u_t^T P_{t-1} u_t}$

## 1. Conjugate Gradient Descent

If the gradient is zero and the hessian is positive definite, then it is a strict relative minimum

Solving quadriatic optimization -> gradient=0 -> this is a linear system

Sometimes we optimize via steepest descent(gradient descent).

Initialization:

$r_0 = b - A\theta_0$ (residual) $c_0 = r_0$ (conjugate gradient direction)

Calculation of the improved solution $\theta_{k+1}$ and the new residual $r_{k+1}$:

$\alpha_k = \frac{r_k^T r_k}{c_k^T A c_k}$
$\theta_{k+1} = \theta_k + \alpha_k c_k$
$r_{k+1} = c_k - \alpha_k A c_k$

Calculation of the new conjugate gradient direction $c_{k+1}$:

$\beta_k = -\frac{r_{k+1}^T A c_k}{c_k^T A c_k}$
$c_{k+1} = r_{k+1} + \beta_k c_k$

## 2. Analysis

This method finds the solution for a linear system in exactly m steps. However, this method is ill-conditioned(not robust to noise). In practice, RLS is much more robust.

# 3. Multi-output Weighted Least Squares

## 1. Problem Formulation

The recursive least squares problem can be extended to have a vector output at time t:

$$\bar{y}(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y(t)_l \end{bmatrix} \in \mathcal{R}^l$$

The whole output sequence

$$\hat{y}(t) = \begin{bmatrix} \bar{u}_1^T \\ \vdots \\ \bar{u}_l^T \end{bmatrix} \bar{\theta} = U^T(t)\bar{\theta} \in \mathcal{R}^{l \times m}$$

The residual error:

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

The optimization problem: $\hat{\theta} = \arg\min J_t(\theta)$

The objective function: $J_t(\theta) = \sum_{i=1}^{t} \bar{e}(i)^T W \bar{e}(i)$

Where $W$ is the weight vector.

## 2. Derivation

Similar to the vanilla RLS,

$\hat{\theta} = P_t Bt$

where

$P_t = [\sum_{i=1}^{t} U^T(i) W U(i)]^{-1}$

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

The recursion is given by:

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

$P_t = P_{t-1} - P_{t-1}U(t)[W^{-1} + U(t)P_{t-1}U(t)]^{-1}U(t)P_{t-1}$

# 4. Kalman Filters

## 1. Introduction

We want to estimate some unobservable latent system states using observations under noise. Kalman filters do this.

Using the state model and the observations up to the previous time step, we can predict the current state(a priori). Then, perform an observation, and correct the current estimate using the observation, state model, and the gain.

The difference between the previous estimate and the measurement is amplified by the gain, then added up to the previous estimate to produce the current new estimate(corrects).

The gain K is formulated in a certain way that it balances the significance of the previous estimate and the measured value.

If gain is big: measurements are accurate, estimates are unstable

If gain is small: measurements are inaccurate, estimates are stable(small error)

typically, the gain gets smaller w.r.t. time.

The RLS algorithm is a subset of Kalman filters.

### Standard Procedure

The standard procedure:

1. Calculate the gain
2. Calculate the current estimate
3. Calculate the new estimate error
4. Update the estimate and the model

## 2. Problem Formulation

### 1. The Model

We consider the problem where we want to know the unobservable state of some system. Since the state is unobservable, we cannot measure it directly. However, we can make indirect observations $y_t$.

The true state at time t+1 $x_{t+1}$ is given by:

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

Where $A_t$ is the state transition matrix, $B_t$ is the control input model, $u_t$ is the control signal, $G_t$ is the noise matrix, and $w_t$ is the process noise.

The obeservations are given by:

$$y_t = H_t x_t + v_t$$

Where $H_t$ is the observation model and $v_t$ the observation noise.

### 2. Noise Assumptions

We assume that the process noise and the measurement noise have zero mean(if not so then we can just subtract it from the whole thing). Also, we assume that the noises measured at different time points are uncorrelated.

$E[w_t] = 0$, $E[v_t] = 0$

$C_v(t, s) = E[v_t v_s^T] \in \mathcal{R}^{l \times l}$

$C_w(t, s) = E[w_t w_s^T] \in \mathcal{R}^{n \times n}$

$C_{vw}(t, s) = 0$

$$C_v(t, s) = \begin{cases} 0, & \forall t \neq s \\ R_t, & t = s \end{cases}$$

$$C_w(t, s) = \begin{cases} 0, & \forall t \neq s \\ Q_t, & t = s \end{cases}$$

## 3. The objective Function

To drive the hidden state estimate to the true state, we minimize the expected mean squared error of the estimated state and the unobservable actual state:

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

This is done by the Kalman Filter algorithm described below.

## 4. Algorithm Outline

#### 1. Time update prediction

1. Make a state estimate $\hat{x}_{t-1}$ using all knowledge up to t-1.
2. Make a priori estimate by getting the expected state based on the state transition model $\hat{x}_{t-1} \Rightarrow x_{t|t-1}$.

#### 2. Measurement update correction

1. Calculate the estimation error $y_t - H_t \hat{x}_t$.
2. Make correction to the state machine.
3. Make a posteriori estimate $\hat{x}_t$.

## 5. State Transition

Expected state transition: $E[x_{t+1}] = A_t E[x_t]$

Projection(prediction): $\hat{x}_{t|t-1} = A\hat{x}_{t-1}$

$\hat{y}_t = H\hat{x}_{t|t-1}$

Transition: $\hat{x}_t = \hat{x}_{t|t-1} + K_t(y_t - H_t\hat{x}_t)$

Where $\hat{x}_{t|t-1}$ is the predicted(a priori) state estimate predicted using the information up to t-1. $\hat{x}_t$ is the a posteriori estimate updated with the current observation $y_t$.