CARLETON UNIVERSITY

SCHOOL OF
MATHEMATICS AND STATISTICS

HONOURS PROJECT

TITLE: Recursive and Distributed Least Square Algorithms in Linear Regression Models

AUTHOR: Siyue Sun

SUPERVISOR: Minyi Huang

DATE: August 23, 2020

Contents

1 Introduction

In statistics, regression is a useful technique for modelling the relationship between two or more variables. The regression analysis allows you to build a regression equation and make a prediction [7]. In this paper, we will only focus on predicting the linear relationship between response (output) variable and predictor (input) variable.

The earliest form of regression is the method of least square [7]. Legendre proposed the first elaborate exposition of the least square method in 1805. This is the most natural approach to investigate the effects of one variable to another. The procedure can help to find the best fit of the given data by minimizing the sum of squared residuals. Besides that, a best-fitted line can determine whether or not our data has a linear relationship. Moreover, if we receive data sequentially, we do not need to resolve our least square estimates every time. Alternatively, we can try to apply the recursive least square algorithms, and this algorithm was discovered by Gauss(1821). It updates an estimator recursively of the given measurement at the current stage. Recently, the distributed algorithms over the network has been widely used in various practical application. We will propose a distributed least square algorithm over a network for multi-agent[1][3].

This paper is summarized as follows, in section 2, I will introduce the background of the regression model, the non-recursive least square analysis, and

recursive least square analysis. In section 3, we define the autoregressive model and conduct a least square estimation with the model[4]. The remaining part of the section will demonstrate the distributed least square algorithm over a network.

2 Non-recursive and Recursive Least Square Estimation

2.1 Basic Regression Model

The regression model usually has the form:

$$y_i = b_0 + b_1 x_{i1} + b_2 x_{i2} + ... + b_p x_{ip} + \epsilon_i, i = 1, 2, ..., n \tag{2.1.1}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_p \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

where y_i is i^{th} response variable; x_i is i^{th} predictor variable; b_i is an estimated coefficient of the independent variable; ϵ_i is a random error term, with mean zero and variance σ^2.

The equation(2.1.1) can be written in a matrix form:

$$Y = X\beta + \epsilon \tag{2.1.2}$$

where $X \in \mathbb{R}^{(n+1)\times(p+1)}$ is the predictor vector; $Y \in \mathbb{R}^{n\times 1}$ is the response vector; $\beta \in \mathbb{R}^{p\times 1}$ is the unknown parameter vector to be estimated; $\epsilon \in \mathbb{R}^{n\times 1}$ is the random error vector.

2.2 Non-Recursive Least Square(LS)

simple linear regression in regular form

The parameter vector β can be estimated by the non-recursive least square method. Let us consider the simple linear regression model (2.1.2)

$$\beta = \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}, X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$$

The simple linear regression has the form:

$$y_i = b_0 + b_1 x_i + \epsilon_i \tag{2.2.1}$$

where b_0 and b_1 are the parameters of the model; y_i is the i^{th} response from subject; x_i is the i^{th} predicted value from subject.

In simple linear regression, our goal is to predict y_i from the knowledge of x_i based on the model (2.1.2). The simplest method is the non-recursive least squares method, and the approach will minimize

$$\epsilon = min||y_i - (b_0 + b_1 x_i)|| \tag{2.2.2}$$

Suppose we have n pairs of data (x_1,y_1), (x_2,y_2) ...(x_n,y_n) and unknown parameter b_0, b_1.

We can use non-recursive least square method to obtain parameters b_0 and b_1. Based on the fitted line $b_0 + b_1 x_i$, we can define a function called "residual sum of squares"(RSS).

$$RSS = \sum_{i=1}^{n} (y_i - (b_0 + b_1 x_i))^2 \tag{2.2.3}$$

The least squares method obtains b_0 and b_1, so that residual sum of squares is minimized. b_0 and b_1 can be obtained by taking derivatives of residual sum of squares.

$$\frac{\partial RSS}{\partial b_0} = -2 \sum_{i=1}^{n} (y_i - (b_0 + b_1 x_i)) = 0 \tag{2.2.4}$$

$$\frac{\partial RSS}{\partial b_1} = -2 \sum_{i=1}^{n} (x_i (y_i - (b_0 + b_1 x_i)) = 0 \tag{2.2.5}$$

After some calculations, the least squares estimators $\hat{b}_0$ and $\hat{b}_1$ are

$$\hat{b}_1 = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sum_{i=1}^{n} {x_i}^2 - n\bar{x}^2} = \frac{S_{xy}}{S_{xx}} \tag{2.2.6}$$

$$\begin{aligned} \hat{b}_0 &= \frac{\sum_{i=1}^{n} y_i}{n} - \hat{b}_1 \frac{\sum_{i=1}^{n} x_i}{n} \\ &= \bar{y} - \hat{b}_1 \bar{x} \end{aligned} \tag{2.2.7}$$

The estimated least squares fitted line $\hat{y}_i$ becomes:

$$\begin{aligned} \hat{y}_i &= \hat{b}_0 + \hat{b}_1 x_i \\ &= \bar{y} + \hat{b}_1 (x_i - \bar{x}) \end{aligned} \tag{2.2.8}$$

Multiple regression in matrix form

The parameters β can be estimated by establishing the strong consistency of least squares. Let's solve the least squares for the multiple regression model.

$$y_i = b_1 x_{i1} + b_2 x_{i2} + ... + b_p x_{ip} + \varepsilon_i \ \ (i = 1, 2, ...) \tag{2.2.9}$$

Remark: The basic regression model has the general form (2.2.9), but usually x_{i1} is 1. Let's consider the general form, and the β can also be estimated by using matrix form.

$$\hat{\beta} = (X^t X)^{-1} X^t Y \tag{2.2.10}$$

with the estimate of variance $\hat{\beta}_j$:

$$v(\hat{\beta}_j) = \sigma^2(c_{ii})^2 \tag{2.2.10}$$

where c_{ii} is the i^{th} value of the diagonal element on $(X^tX)^{-1}$; σ is the variance of the noise.

To establish the strong consistency of least squares of multiple regression model, the following assumptions should be satisfied[5]:

1. $\sum_{i=1}^{\infty} c_i\varepsilon_i$ converges as. for all real sequences ci s.t $\sum_{i=1}^{\infty} {c_i}^2 < \infty$

2. X^t is non-singular and $\varepsilon_i \overset{iid}{\sim} NID(0, \sigma^2)$

Remark: NID stands for normally and independently distributed.

If we calculate the parameter β using Equation 2.2.9:

$$\begin{aligned}\hat{\beta} =&(X^tX)^{-1}X^tY \\ =&(\begin{bmatrix} x_{11} & x_{21} & \cdots & x_{n1} \\ \vdots & \vdots & \vdots & \vdots \\ x_{1p} & x_{2p} & \cdots & x_{np} \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix})^{-1} \begin{bmatrix} x_{11} & x_{21} & \cdots & x_{n1} \\ \vdots & \vdots & \vdots & \vdots \\ x_{1p} & x_{2p} & \cdots & x_{np} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{bmatrix}\end{aligned}$$

Example of non-recursive least square estimation

Suppose we are given the following data

$$X = \begin{bmatrix} 0 & 1 \\ 2 & 0 \\ 3 & 0 \\ 2 & 0 \\ 4 & 0 \\ 1 & 0 \end{bmatrix}, Y = \begin{bmatrix} -1 \\ 2 \\ 4 \\ 1 \\ 4 \\ 5 \end{bmatrix},$$

Find the least square regression line that indicates the relationship between X and Y.

Solution:

The parameter β can be estimated by

$$\begin{aligned} \hat{\beta} &= (\frac{1}{n}\sum_{i=1}^{n} x_i {x_i}^t)^{-1}(\frac{1}{n}\sum_{i=1}^{n} {x_i}^t y_i) \\ &= \begin{bmatrix} 1.1471 \\ -1.0000 \end{bmatrix} \end{aligned}$$

The fitted line is

$$Y = \begin{bmatrix} 1.1471 \\ -1.0000 \end{bmatrix} X$$

However, Y can not be calculated from the fitted line.

The predicted values are

$\hat{y}_i = b_1 x_{i1} + b_2 x_{i2}$

The i^{th} residual ϵ_i is defined as

$$\epsilon_i = y_i - \hat{y}_i$$

Using MATLAB

$$\epsilon = \begin{bmatrix} -1 \\ 0.2948 \\ 0.4422 \\ 0.2948 \\ 0.5884 \\ 0.1471 \end{bmatrix}$$

Finally, Y can be produced by the following equation

$$Y = \begin{bmatrix} 1.1471 \\ -1.0000 \end{bmatrix} X + \begin{bmatrix} -1 \\ 0.2948 \\ 0.4422 \\ 0.2948 \\ 0.5884 \\ 0.1471 \end{bmatrix}$$

2.3 Recursive Least Square(RLS)

In reality, suppose we have already calculated t-1 sets of data, but new data is coming in, and we have to recalculate the data over and over. In this situation, if we still use least square algorithms, it would be time-consuming. We want to know how to recursively update the least squares estimate of an estimator. Refer to Simon's book [2], he introduces a method of so-called Recursive Least Square algorithm. Under this algorithm, we can update estimate recursively with each new measurement, so we do not waste time on recomputing everything each time a new data comes in. We first express the relation:

$$\begin{aligned} y_1 &= H_{11}x_1 + ... + H_{1n}x_n + v_1 \\ &\vdots \\ y_m &= H_{m1}x_n + ... + H_{mn}x_n + v_n \end{aligned}$$

where y_a is the a^{th} element noisy measurement; x_b is the b^{th} unknown element to be estimated; v_c is the c^{th} element of some measurement noise. Our goal is to find the best estimate $\hat{x}$ of x. This system of linear equations can be represented in matrix form:

$$Y = H_k X + V \tag{2.3.1}$$

where $X \in \mathbb{R}^{n\times 1}$ is an unknown constant vector; $Y \in \mathbb{R}^{m\times 1}$ is the noisy measurement vector; $V \in \mathbb{R}^{n\times 1}$ is a measurement noise vector; H $\in \mathbb{R}^{m\times n}$ is a $m \times n$ matrix.

Suppose we have $k-1$ sets of data and a new measurement y_k comes in, we want to find the best estimate $\hat{x}_k$ on the basis of the previous estimate $\hat{x}_{k-1}$. We shall define a linear recursive estimator as [2]

$$y_k = H_k X + v_k \tag{2.3.2}$$

$$\hat{x}_k = \hat{x}_{k-1} + K_k(y_k - H_k\hat{x}_{k-1}) \tag{2.3.3}$$

where $K_k \in \mathbb{R}^{n\times m}$ is a estimator gain matrix; $y_k - H_k\hat{x}_{k-1}$ is a correlation term.

Method A

Recursive least squares estimation can be summarized as follows [2]:

1. Initial the estimator

$$\hat{x}_0 = E(x)$$
$$p_0 = E[(x - \hat{x_0})(x - \hat{x_0})^T]$$

If we do not have any information about x, set $P_0 = \infty I$. If we have perfectly knowledge about x, then set $P_0 = 0$.

2. Iteratively improve our estimate by doing the following:

a) obtain the new measurement y_k by assuming it has the form

$$y_k = H_k x + v_k \tag{2.3.4}$$

where v_k is a random vector with zero mean with covariance R_k.

Assume the measurement noise at each time step k is independent, that is, $E(v_1 v_k) = R_k \delta_{k-i}$.

where δ_k is the dirac function.

b) Update the estimate $\hat{x}_k$ and estimation error recursively by:

$$\begin{aligned} K_k &= P_{k-1} {H_k}^T (H_k P_{k-1} {H_k}^T + R_k)^{-1} \\ P_k &= I - K_k H_k P_{k-1} (I - K_k H_k)^T + K_k R_k {K_k}^T \\ \hat{X}_k &= \hat{X}_{k-1} + K_k (y_k - H_k \hat{X}_{k-1}) \end{aligned}$$

Method B

For the recursive least square algorithm that we stated above, there is another way of estimating the best estimate of θ. We shall introduce the stochastic regression model:

$$\begin{aligned} Y =& {x_t}^T \theta + \varepsilon_n \\ =& x_1 \theta_1 + x_2 + \cdots + x_n \theta_n + \varepsilon_t \end{aligned} \tag{2.3.5}$$

where $Y \in R^{n\times 1}$ is the output vector sequence; $x_t \in R^{1\times n}$ is the input vector sequence; $\theta \in R^{n\times 1}$is an unknown parameter to be estimated; ε_t is the disturbance vector sequence.

The best estimate of $\hat{\theta}$ can be updated recursively by [6]:

$$\begin{aligned} \hat{\theta}_t &= \hat{\theta}_{t-1} + P_t x_t(\hat{\theta}_{t-1} x_t), \\ P_t &= P_{t-1} - \frac{P_{t-1} x_t {x_t}^T P_{t-1}}{1 + {x_t}^T P_{t-1} x_t} \end{aligned} \tag{2.3.6}$$

In this method, R_k has been eliminated and algorithm is shorter than method A.

Example of Recursive least square:

Suppose the measurement noise v_k has a zero mean and a variance $R_k = 0.02$. The measurement equation is given as follows

$$y_k = [k \;\; 0.79 * (k-1)]x + v_k$$

and the true estimates are $x_1 = 30$, $x_2 = 20$.

Solution:

H can be obtained from the measurement equation:

$$H = \begin{bmatrix} 1.0000 & 0 \\ 2.0000 & 0.7900 \\ 3.0000 & 1.5800 \\ 4.0000 & 2.3700 \\ 5.0000 & 3.1600 \\ 6.0000 & 9.9500 \end{bmatrix}$$

If we solve the question using **method A**:

1. we know the true value of x before measurement, take $P_0 = 0$.

2. update our estimate recursively using equation (2.b)

   when k=1

$$K_1 = P_1 {H_1}^T (H_1 P_{K-1} {H_1}^T + R_1)^{-1} = \begin{bmatrix} 0.9804 \\ 0 \end{bmatrix}$$

$$\hat{x}_1 = \hat{x}_0 + K_1(y_1 - H_1 \hat{x}_0) = \begin{bmatrix} 30.577 & 19.000 \end{bmatrix}$$

$$P_1 = (I - K_1 H_1) P_0 = \begin{bmatrix} 0.0196 & 0 \\ 0 & 1 \end{bmatrix}$$

when k=2

$$K_2 = P_2 {H_2}^T (H_1 P_{K-2} {H_2}^T + R_2)^{-1} = \begin{bmatrix} 0.0543 \\ 1.0934 \end{bmatrix}$$

$$\hat{x}_2 = \hat{x}_1 + K_1(y_2 - H_2\hat{x}_1) = \begin{bmatrix} 30.0875 \\ 19.6003 \end{bmatrix}$$

$$P_2 = (I - K_2 H_2)P_1 = \begin{bmatrix} 0.0175 & -0.0429 \\ -0.0429 & 0.1362 \end{bmatrix}$$

when k=3

$$K_3 = P_3 {H_3}^T (H_2 P_{K-3} {H_3}^T + R_3)^{-1} = \begin{bmatrix} -0.1380 \\ 0.7808 \end{bmatrix}$$

$$\hat{x}_3 = \hat{x}_1 + K_2(y_3 - H_3\hat{x}_2) = \begin{bmatrix} 30.0305 \\ 19.8995 \end{bmatrix}$$

$$P_3 = (I - K_3 H_3)P_2 = \begin{bmatrix} 0.0154 & -0.0309 \\ -0.0309 & 0.0686 \end{bmatrix}$$

when k=4

$$K_4 = P_3 {H_4}^T (H_4 P_{K-3} {H_4}^T + R_4)^{-1} = \begin{bmatrix} -0.1823 \\ 0.5995 \end{bmatrix}$$

$$\hat{x}_4 = \hat{x}_3 + K_2(y_4 - H_4 \hat{x}_3) = \begin{bmatrix} 30.0305 \\ 19.9131 \end{bmatrix}$$

$$P_4 = (I - K_4 H_4) P_3 = \begin{bmatrix} 0.0132 & -0.0238 \\ -0.0238 & 0.0453 \end{bmatrix}$$

when k=5

$$K_5 = P_4 {H_5}^T (H_5 P_4 {H_5}^T + R_5)^{-1} = \begin{bmatrix} -0.1879 \\ 0.4854 \end{bmatrix}$$

$$\hat{x}_5 = \hat{x}_4 + K_5(y_5 - H_5 \hat{x}_4) = \begin{bmatrix} 29.9994 \\ 19.9933 \end{bmatrix}$$

$$P_5 = (I - K_5 H_5) P_4 = \begin{bmatrix} 0.0115 & -0.0193 \\ -0.0193 & 0.0337 \end{bmatrix}$$

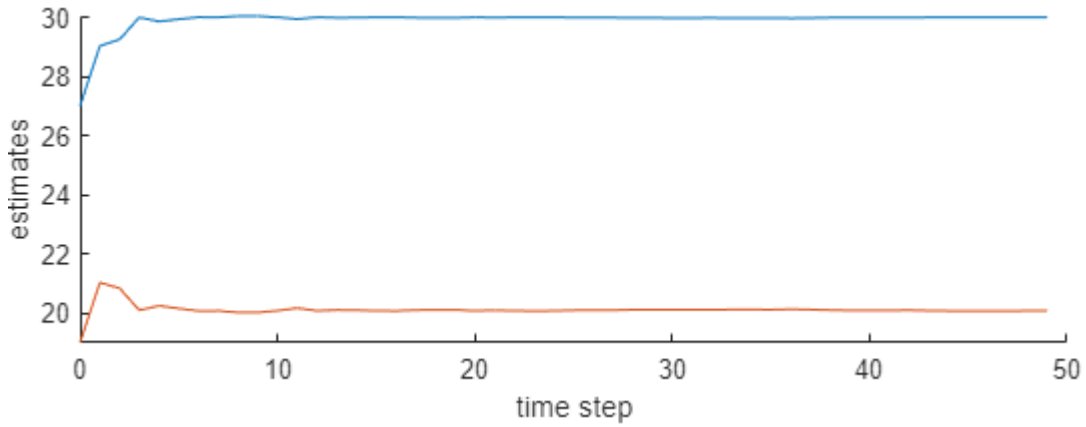
After 5 updates, the final estimates

$$\hat{x}_5 = \begin{bmatrix} 29.9994 \\ 19.9933 \end{bmatrix}$$

is getting close to

$$x = \begin{bmatrix} 30 \\ 20 \end{bmatrix}$$

On the simulation below around time step 12, the final estimate $\hat{x}_1$(blue line) converge to the true estimates x_1= 20, and the final estimate $\hat{x}_2$(red line) converge to the true estimate x_2.



3 Distributed Least Square Algorithm

3.1 Autoregressive model

As discussed in the last section about the recursive least squares algorithm, the RLS algorithm recursively updates the estimate of a parameter. In this section, we will compare the autoregressive model with the distributed least square algorithm. The autoregressive model is a time series model that uses

observation from previous time steps to predict the value at the present stage. In general, an autoregressive model can predicts current value of the series, x_t , based on a function of past values p, $x_{t-1}, x_{t-2}, ...x_{t-p}$. where p is the order of the AR model.

An AR(P) model can be defined as [4]:

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + ... + \phi_p x_{t-p} + \omega_t \tag{3.1.1}$$

where ω_t is a white noise with mean zero and variance σ^2

Remark: ω_t is independent form the future.

We can investigate the simplest form of the AR model AR(1), as the following:

$$x_t = \phi_1 x_{t-1} + \omega_t, |\phi_1| < 1 \tag{3.1.2}$$

If we iterate backwards k times, we get

$$\begin{aligned} x_t &= \phi_1 x_{t-1} + \omega_t \\ &= \phi_1(\phi_1 x_{t-2} + \omega_{t-1}) + \omega_t \\ &= {\phi_1}^2 + \phi_1 \omega_{t-1} + \omega_t \\ &\vdots \\ &= {\phi_1}^k x_{t-k} + \sum_{j=0}^{k-1} {\phi_1}^j \omega_{t-j} \end{aligned} \tag{3.1.3}$$

when k is huge ${\phi_1}^k x_{t-k} = 0$, we can represent an AR(1) model as a linear process:

$$x_t = \sum_{j=0}^{k-1} {\phi_1}^j \omega_{t-j} \tag{3.1.4}$$

The equation (3.1.4) is known as stationary AR(1), and the mean of the model can be defined as

$$E(x_t) = \sum_{j=0}^{\infty} E(\omega_{t-j}){\phi_1}^j = 0 \tag{3.1.5}$$

and the variance is

$$\begin{aligned} var(x_t) &= E((x_t)^2) - (\mu)^2 \\ &= \frac{\sigma_\epsilon^2}{1-\varphi^2} \end{aligned} \tag{3.1.6}$$

Example of LS estimation in AR(2) process

Suppose we want to simulate AR(2) process in order to find least square estimate θ.

where

$$\theta \in (\hat{\phi_1}\hat{\phi_2})^T = (X^T X)^{-1} X^T Y$$

We specify the AR(2) model as:

$$Y(t) = \theta_1 Y(t-1) - \theta_2 Y(t-2) + \varepsilon(t) \; where \quad \varepsilon(t) \in (0, \sigma^2)$$

Suppose we have 5 data: $y_1 = 1.2, y_2 = 3.2, y_3 = 2.4, y_4 = 2.2, y_5 = 2.3$

$\hat{\theta}$ can be estimated using equation [8]:

$$\hat{\theta} \in (\phi_1\ \phi_2)^T = (X^T X)^{-1} X^T Y$$
$$= \left(\begin{bmatrix} \sum_{i=2}^{n-1} Y_i^2 & \sum_{i=2}^{n-1} Y_i Y_{i-1} \\ \sum_{i=2}^{n-1} Y_i Y_{i-1} & \sum_{i=1}^{n-2} Y_i^2 \end{bmatrix} \right)^{-1} \begin{bmatrix} -\sum_{i=2}^{n-1} Y_i Y_{i+1} \\ -\sum_{i=1}^{n-2} Y_i Y_{i+2} \end{bmatrix}$$

Find the best estimate of θ.

Solution:

we have observation $y_1 = 1.2, y_2 = 3.2, y_3 = 2.4, y_4 = 2.2, y_5 = 2.3$

and

$$X = \begin{bmatrix} -y_2 & -y_1 \\ -y_3 & -y_2 \\ -y_4 & -y_3 \end{bmatrix} = \begin{bmatrix} -3.2 & -1.2 \\ -2.4 & -3.2 \\ -2.2 & -2.3 \end{bmatrix}$$

Using MATLAB we have the result:

$$\begin{aligned}\hat{\theta} = \begin{bmatrix}\hat{\phi_1}\\ \hat{\phi_2}\end{bmatrix} &= (X^T X)^{-1} X^T Y \\ &= (\begin{bmatrix}\sum_{i=2}^{n-1} Y_i^2 & \sum_{i=2}^{n-1} Y_i Y_{i-1}\\ \sum_{i=2}^{n-1} Y_i Y_{i-1} & \sum_{i=1}^{n-2} Y_i^2\end{bmatrix})^{-1} \begin{bmatrix}-\sum_{i=2}^{n-1} Y_i Y_{i+1}\\ -\sum_{i=1}^{n-2} Y_i Y_{i+2}\end{bmatrix} \\ &= (\begin{bmatrix}20.84 & 16.8\\ 16.8 & 17.44\end{bmatrix})^{-1} \begin{bmatrix}-18.02\\ -15.44\end{bmatrix} \\ &= \begin{bmatrix}-0.6757\\ -0.2344\end{bmatrix}\end{aligned}$$

Therefore, our AR(2) model has the form:

$$X(t) = -0.6757X(t-1)0.2344X(t-2) + \varepsilon(t) \; where \quad \varepsilon(t) \in (0, \sigma^2)$$

3.2 The distribution of the LS algorithm over a network

In this section, we will express the distributed least square estimation over a sensor network [1][3]. The difference between the standard and distributed least square algorithms is distributed algorithms can break complicated computation problem into many small clusters and then solve it in a distributed manner. The algorithm allows one to reach the corresponding solution to

the overall set cooperatively. Let's consider a network of N agents, and each agent can communicate with a specific agent with the same information. If each agent wants to achieve the same least-squares solution, the problem is equivalent to find the minimization of the cost function:

$$J = (x - H\theta)^T (x - H\theta) \tag{3.3.1}$$

where x is a $p \times 1$ observation vector; H is a $p \times q$ observation matrix; θ is a $q \times 1$ unknown parameter vector to be estimated.

If we set the gradient of equation (3.3.1) to zero, our linear least square estimator can be obtained as[1]:

$$\hat{\theta}_{LLS} = (H^T H)^{-1}(H^T)x. \tag{3.3.2}$$

$$H = \begin{bmatrix} (h_1)^T \\ (h_2)^T \\ \vdots \\ (h_N)^T \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

where $(h_i)^T$ is the i^{th} row vector of the matrix H.

We want to take advantage of distributed linear least square estimation to obtain an optimal result over multi-agent networks, we first introduce some basic concepts of algebraic graph theory [1].

The graph $G = (N, \varepsilon)$ consists a finite set of nodes $N = 1, 2, 3, ...$, and a set of edges ε.

Definition: The adjacency matrix A, is an $N \times N$ binary matrix defined as following

$$A_{ij} = \begin{cases} 1, & \text{if vertex i is adjacent to vertex j} \\ 0, & \text{otherwise} \end{cases}$$

Definition: An arc (i, j) is said to be incident to each of the nodes i and j if the incidence matrix B_{ie} follows:

$$B_{ie} = \begin{cases} 1, & \text{if node i and arc (i, k) are incident} \\ 0, & \text{otherwise} \end{cases}$$

Definition: The Laplacian matrix is an $N \times N$ matrix defined as:

$$L = D - A$$

where D is a diagonal matrix; L is a symmetric positive semi-definite matrix.

To obtain an updated algorithm for our distributed least square estimator, we should introduce symmetric weighted matrix W. For an undirected graph,

(1) W is a diagonal $E \times E$ edge-weight matrix s.t $W_{ee} > 0$ associated with the undirected graph G.

(2) The weighted Laplacian matrix is denoted by $\bar{L}$ with the relation of W as $\bar{L} = BWB^T$

Method A

Considered the following iteration for the i^{th} sensor [1]:

$$x_i(k+1) = x_i(k) + \delta \sum_{j:e=(i,j)\in\varepsilon} W_{ee}(x_j(k) - x_i(k)), \delta \in (0,1) \tag{3.3.3}$$

where $x_i(k+1)$ is the state of node k+1 at iteration i.

We can express the equation (3.3.3) in a matrix form[1]:

$$\begin{aligned} x(k+1) &= x(k) - \Delta(BWB^T)x(k) \\ &= (I_N - \Delta\tilde{L})x(k) \\ &= My(k) \end{aligned} \tag{3.3.4}$$

where I_N is $N \times N$ identity matrix; M is a symmetric matrix.

Method B

We can solve the distributed least square algorithm over an undirected network in a different method. To distinguish these two methods, we use different notation:

$$\theta_{lls} = (H^T H)^{-1} H^T z$$

Three assumptions[3] need to be satisfied before using the distributed least square algorithm for updating each agent of the information from its neighboring agent.

Assumption 1 : Assume matrix H has full column rank i.e $rank(H) = m$ where H $\in R^{N\times m}$.

Assumption 2 : Assume the interaction graph G is undirected and connected.

Assumption 3 : Assume the mixing weighted N $\times$ N matrix W has the following two properties:

(a) For any (i,j)$\in \varepsilon$, $w_{ij} > 0$. Besides, $w_{ii} > 0$ for all i $\in$ N.

(b) The matrix W is symmetric and doubly stochastic, which means $W^T = W$ and the sum of each row and each column all equal to 1.

Consider the following iteration, each node i can update its state vectors at each stage t as[3]:

$$\begin{aligned} x_i(t+1) &= \sum_{j\in N_i} W_{ij}x_j(t) - \alpha v_i(t), \\ v_i(t+1) &= \sum_{j\in N_i} W_{ij}v_j(t) + \bigtriangledown f_i(x_i(t+1)) - \bigtriangledown f_i(x_i(t)) \end{aligned} \tag{3.3.5}$$

where $\alpha > 0$ is the step size; $\bigtriangledown f_i()$ is the gradient of the local object function $f_i()$.

The initial condition $x_i(0)$ can be chosen arbitrarily, and $v_i(0) = \bigtriangledown f_i(x_i(0))$, and

$$\bigtriangledown f_i(x_i(t)) = h_i {h_i}^T x_i(t) - z_i h_i \tag{3.3.6}$$

where z_i is the observation vector; h_i^T is the i^{th} row vector of the observation matrix H. Let's rewrite the algorithm (3.3.5 and 3.3.6) in a matrix form [3]:

$$\begin{aligned} x(t+1) &= (W \otimes I_m)x(t) - \alpha v(t), \\ v(t+1) &= (W \otimes I_m)v(t) + \bigtriangledown F(x(t+1)) - \bigtriangledown F(x(t)) \end{aligned} \tag{3.3.7}$$

Remark: "$\otimes$" is called the tensor product. If we have two vectors space $V \in R^N$ and $W \in R^M$, then the tensor product of two $V \otimes W \in R^{N \times M}$.i.e

$$V = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}, W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}, V \otimes W = \begin{bmatrix} v_1 w_1 & v_1 w_2 & \cdots & v_1 w_m \\ v_2 w_1 & v_2 w_2 & \cdots & v_2 w_m \\ \vdots & \vdots & \vdots & \vdots \\ v_n w_1 & v_n w_2 & \cdots & v_n w_m \end{bmatrix}$$

The initial condition $v(0) = \bigtriangledown F(x(0))$, $\alpha > 0$ is the step size, and

$$\bigtriangledown F(x(t)) = \tilde{H} x(t) - z_H \tag{3.3.8}$$

where

$$x(t) = [{x_1}^T, (t){x_1}^T(t), \cdots, {x_N}^T(t)]^T$$

$$v(t) = [{v_1}^T(t), {v_2}^T(t), \cdots, {v_N}^T(t)]^T$$

$$\tilde{H} = diag(h_1{h_1}^T, h_2{h_2}^T, \cdots, h_N{h_N}^T)$$

$$z_H = [z_1{h_1}^T, z_2{h_2}^T, \cdots, z_N{h_N}^T]^T$$

The solution of distributed least square algorithm will exponentially converge to the least square solution if and only if $\alpha < \bar{\alpha}$, where[3]

$$\bar{\alpha} = \frac{1}{2\lambda_{max}\big[(I_N + W)^{-2} \otimes I_m\big]\tilde{H}} \tag{3.3.9}$$

where $\lambda_{max}\big[L\big]$ means the largest eigenvalue of the Laplacian matrix L.

Example of distributed LS algorithm of six-node network

Suppose we have an undirected graph G with six nodes(agents) with the following information:

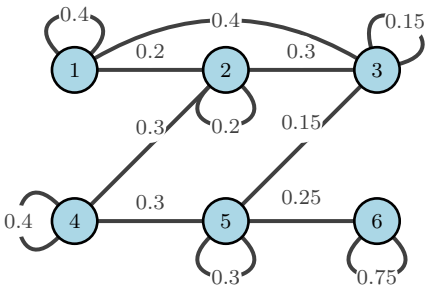


figure 1

$$H = \begin{bmatrix} 0 & 1 \\ 2 & 0 \\ 3 & 0 \\ 2 & 0 \\ 4 & 0 \\ 1 & 0 \end{bmatrix}, z = \begin{bmatrix} -1 \\ 2 \\ 4 \\ 1 \\ 4 \\ 5 \end{bmatrix},$$

The initial condition x(0)=$[1, 5, 4, -5, 1, 5, 4, 2, 4, 9, 1, 3]^T$

Find the optimal solution in the sense of the whole network.

Solution:

Let's verify the three assumptions of method B

1. H is a 4×2 matrix with full rank = 2,

2. Based on figure1, it is obvious that graph G is undirected and connected .

3. W is symmetric and $W * 1_N = 1_N$

Since three assumptions hold, we can calculate the unique least square solution by:

$$\hat{\theta}_{LLS} = (H^T H)^{-1}(H^T)z$$

$$= \left(\begin{bmatrix} -1 & 2 & 3 & 2 & 4 & 1 \\ 1 & 2 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 2 & 0 \\ 3 & 0 \\ 2 & 0 \\ 4 & 0 \\ 1 & 0 \end{bmatrix} \right)^{-1} * \begin{bmatrix} 0 & 2 & 3 & 2 & 4 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} -1 \\ 2 \\ 4 \\ 1 \\ 4 \\ 5 \end{bmatrix}$$

$$= \begin{bmatrix} 1.1471 \\ -1.0000 \end{bmatrix}$$

Based on the figure, we can get the mixing weight matrix:

$$W = \begin{bmatrix} 0.4 & 0.2 & 0.4 & 0 & 0 & 0 \\ 0.2 & 0.2 & 0.3 & 0.3 & 0 & 0 \\ 0.4 & 0.3 & 0.15 & 0 & 0.15 & 0 \\ 0 & 0.3 & 0 & 0.4 & 0.3 & 0 \\ 0 & 0 & 0.15 & 0.3 & 0.3 & 0.25 \\ 0 & 0 & 0 & 0 & 0.25 & 0.75 \end{bmatrix}$$

The initial condition v(0) and x(0):

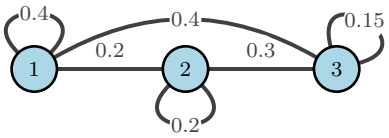
$\mathrm{v}(0) = \tilde{H}x(t) - z_H = [0, 6, 12, 0, -3, 0, 14, 0, 48, 0, -4, 0]^T$

$\mathrm{x}(0) = [1, 5, 4, -5, 1, 5, 4, 2, 4, 9, 1, 3]^T$

$$\bar{\alpha} = \frac{1}{2\lambda_{max}\big[(I_N + W)^{-2} \otimes I_m\big]\tilde{H}} = 0.02827$$

By equation 3.3.9, the solution of distributed least square will exponentially converge to the least square solution if and only if $\alpha < \bar{\alpha}$. If we set $\alpha = 0.0280$, I can update node i=1, 2, 3, 4, 5, 6 and its state vector at stage t = 1 using equation 3.3.5 and 3.3.6.

Node 1 at stage t = 0:



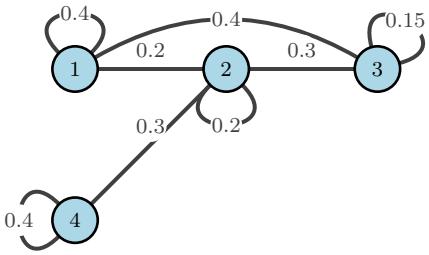
Node 1 receives information from node 2, 3, and itself.

$$
\begin{aligned}
x_1(1) &= \sum_{j \in N_i} W_{ij} x_j(t) - \alpha v_i(t), \\
&= W_{11}x_1(0) + W_{12}x_2(0) + W_{13}x_3(0) - \alpha v_1(0) \\
&= 0.4 \begin{bmatrix} 1 \\ 5 \end{bmatrix} + 0.2 \begin{bmatrix} 4 \\ -5 \end{bmatrix} + 0.4 \begin{bmatrix} 1 \\ 5 \end{bmatrix} - 0.0280 \begin{bmatrix} 0 \\ 6 \end{bmatrix} \\
&= \begin{bmatrix} 1.6 \\ 2.832 \end{bmatrix}
\end{aligned}
$$

State of its neighbor's node and itself.

$$
\begin{aligned}
v_1(1) &= \sum_{j \in N_i} W_{ij} v_j(t) + \bigtriangledown f_i(x_i(t+1)) - \bigtriangledown f_i(x_i(t)) \\
&= W_{11}v_1(0) + W_{12}v_2(0) + W_{13}v_3(0) + h_1 {h^T}_1 x_1(1) - z_1 h_1 + h_1 {h^T}_1 x_1(0) - z_1 h_1 \\
&= \begin{bmatrix} 1.2 \\ 0.232 \end{bmatrix}
\end{aligned}
$$

Node 2 at stage t = 0:



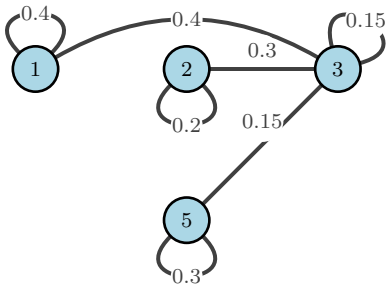
Node 2 receives information from node 1, 3, 4 ,and itself.

$$
\begin{aligned}
x_2(1) &= \sum_{j \in N_i} W_{ij} x_j(t) - \alpha v_i(t), \\
&= W_{21}x_1(0) + W_{22}x_2(0) + W_{23}x_3(0) + W_{24}x_4(0) - \alpha v_2(0) \\
&= 0.2 \begin{bmatrix} 1 \\ 5 \end{bmatrix} + 0.2 \begin{bmatrix} 4 \\ -5 \end{bmatrix} + 0.3 \begin{bmatrix} 1 \\ 5 \end{bmatrix} + 0.3 \begin{bmatrix} 4 \\ 2 \end{bmatrix} - 0.0280 \begin{bmatrix} 12 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} 2.164 \\ 2.1 \end{bmatrix}
\end{aligned}
$$

State of its neighbor's node and itself.

$$
\begin{aligned}
v_2(1) &= \sum_{j \in N_i} W_{ij} v_j(t) + \bigtriangledown f_i(x_i(t+1)) - \bigtriangledown f_i(x_i(t)) \\
&= W_{21}v_1(0) + W_{22}v_2(0) + W_{23}v_3(0) + W_{24}v_4(0) + h_2 {h^T}_2 x_2(1) - z_2 h_2 + h_2 {h^T}_2 x_2(0) - z_2 h_2 \\
&= \begin{bmatrix} -1.644 \\ 1.2 \end{bmatrix}
\end{aligned}
$$

Node 3 at stage t = 0:



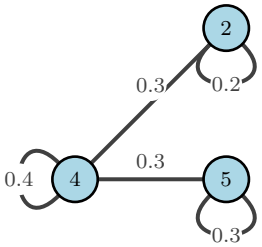
Node 3 receives information from node 1, 2, 5 ,and itself.

$$\begin{aligned} x_3(1) &= \sum_{j \in N_i} W_{ij} x_j(t) - \alpha v_i(t), \\ &= W_{31}x_1(0) + W_{32}x_2(0) + W_{33}x_3(0) + W_{35}x_5(0) - \alpha v_3(0) \\ &= \begin{bmatrix} 2.434 \\ 2.6 \end{bmatrix} \end{aligned}$$

State of its neighbor's node and itself.

$$\begin{aligned} v_3(1) &= \sum_{j \in N_i} W_{ij} v_j(t) + \bigtriangledown f_i(x_i(t+1)) - \bigtriangledown f_i(x_i(t)) \\ &= W_{31}v_1(0) + W_{32}v_2(0) + W_{33}v_3(0) + W_{35}v_5(0) + h_3 {h^T}_3 x_3(1) - z_3 h_3 + h_3 {h^T}_3 x_3(0) - z_3 h_3 \\ &= \begin{bmatrix} 23.256 \\ 2.4 \end{bmatrix} \end{aligned}$$

Node 4 at stage t = 0:



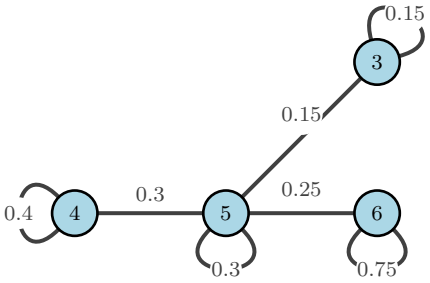
Node 4 receives information from node 2, 5 , and itself.

$$\begin{aligned} x_4(1) &= \sum_{j \in N_i} W_{ij} x_j(t) - \alpha v_i(t), \\ &= W_{42}x_2(0) + W_{44}x_4(0) + W_{45}x_5(0) - \alpha v_4(0) \\ &= \begin{bmatrix} 3.608 \\ 2 \end{bmatrix} \end{aligned}$$

State of its neighbor's node and itself.

$$\begin{aligned} v_4(1) &= \sum_{j \in N_i} W_{ij} v_j(t) + \bigtriangledown f_i(x_i(t+1)) - \bigtriangledown f_i(x_i(t)) \\ &= W_{42}v_2(0) + W_{44}v_4(0) + W_{45}v_5(0) + h_4 {h^T}_4 x_4(1) - z_4 h_4 + h_4 {h^T}_4 x_4(0) - z_4 h_4 \\ &= \begin{bmatrix} 22.032 \\ 0 \end{bmatrix} \end{aligned}$$

Node 5 at stage t = 0:



Node 5 receives information from node 3, 4, 6, and itself.

$$x_5(1) = \sum_{j \in N_i} W_{ij} x_j(t) - \alpha v_i(t),$$

$$= W_{53}x_3(0) + W_{54}x_4(0) + W_{55}x_5(0) + W_{56}x_6(0) - \alpha v_4(0) \quad = \begin{bmatrix} 1.456 \\ 4.8 \end{bmatrix}$$

State of its neighbor's node and itself.

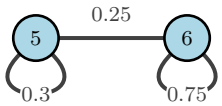
$$v_5(1) = \sum_{j \in N_i} W_{ij} v_j(t) + \bigtriangledown f_i(x_i(t+1)) - \bigtriangledown f_i(x_i(t))$$

$$= W_{53}v_3(0) + W_{54}v_4(0) + W_{55}v_5(0) + W_{56}v_6(0) + h_5 {h^T}_5 x_5(1) - z_5 h_5 + h_5 {h^T}_5 x_5(0) - z_5 h_5$$

$$= \begin{bmatrix} -23.554 \\ 0 \end{bmatrix}$$

Node 6 at stage t = 0:

Node 5 receives information from node 6 and itself.



$$x_6(1) = \sum_{j \in N_i} W_{ij} x_j(t) - \alpha v_i(t),$$

$$= W_{65} x_6(0) + W_{66} x_6(0) - \alpha v_4(0) \quad = \begin{bmatrix} 1.862 \\ 4.5 \end{bmatrix}$$

State of its neighbor's node and itself.

$$v_6(1) = \sum_{j \in N_i} W_{ij} v_j(t) + \bigtriangledown f_i(x_i(t+1)) - \bigtriangledown f_i(x_i(t))$$

$$= W_{65} v_5(0) + W_{66} v_6(0) + h_6 {h^T}_6 x_6(1) - z_6 h_6 + h_6 {h^T}_6 x_6(0) - z_6 h_6$$

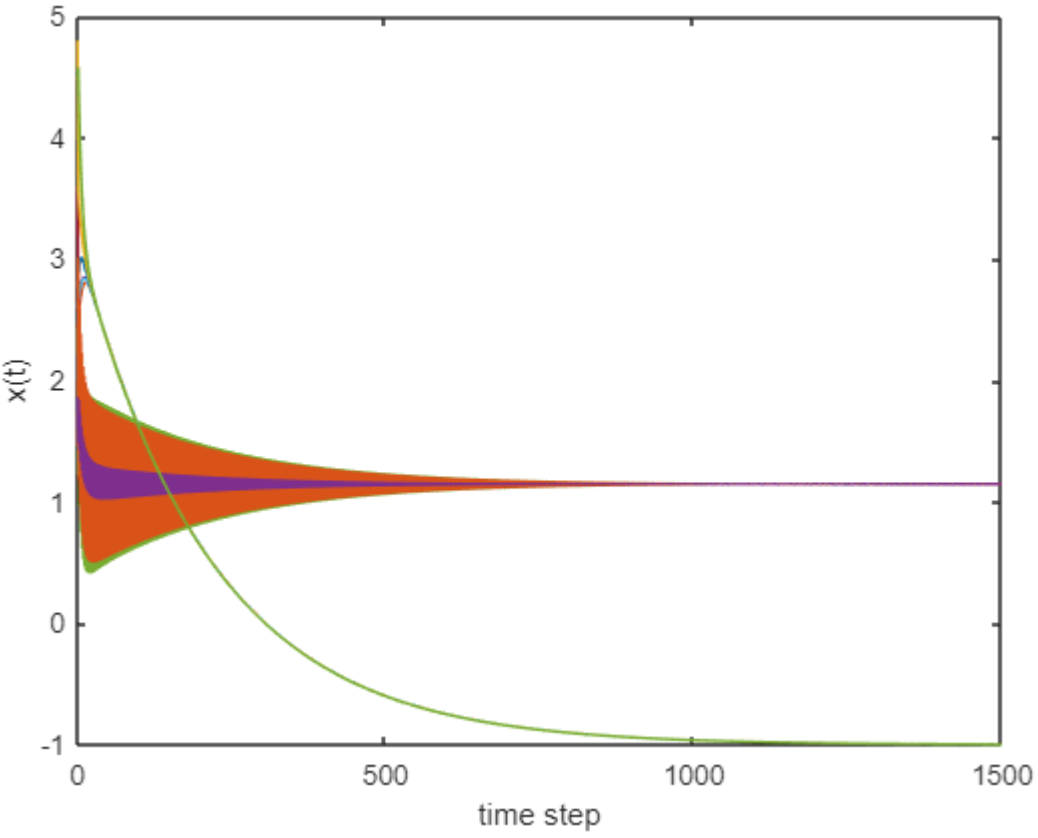
$$= \begin{bmatrix} 9.862 \\ 0 \end{bmatrix}$$

If we use the compact form of the distributed algorithm (3.3.7) to update all nodes at stage 1, the result is the same as the above.

$$x(1) = (W \otimes I_2)x(0) - \alpha v(0) = \begin{bmatrix} 1.6000 \\ 2.8320 \\ 2.1640 \\ 2.1000 \\ 2.4340 \\ 2.6000 \\ 3.6080 \\ 2.0000 \\ 1.4560 \\ 4.8000 \\ 1.8620 \\ 4.5000 \end{bmatrix}$$

$$v(1) = (W \otimes I_2)v(0) + \bigtriangledown F(x(1)) - \bigtriangledown F(x(0)) = \begin{bmatrix} 1.2000 \\ 0.2320 \\ -1.6440 \\ 1.2000 \\ 23.2560 \\ 2.4000 \\ 22.0320 \\ 0 \\ -23.5540 \\ 0 \\ 9.8620 \\ 0 \end{bmatrix}$$

If we repeat the above iteration plenty of times, all six nodes converge exponentially to the least square solution $\hat{\theta}$. As you can see the figure below, all $x_{i,1}(t)$ converge to 1.1471 and all $x_{i,2}(t)$ converge to -1. The result is the same as the least square solution $\theta_{LLS} = [1.1471 \ \ -1.000]^T$



3.3 The comparison of standard and distributed least squares

Compare the example of standard (2.2) and distributed (3.2) least square, we can find that standard least square algorithm is efficient when the dataset

is accessible and small. As we stated the example on section 2.2, we can conduct a standard least square algorithm if one observer can access all the measurements. However, when the dataset is large and strictly private, this becomes complicated to conduct a standard least square algorithm. In this case, we can apply the distributed least square algorithm. The distributed least square algorithm can solve a large amount of data by dividing data into many small pieces of linear equations. The key idea of distributed least square algorithm is each agent limits to update their states and reach a global solution with its states and neighbour's states.

4 Conclusion

This paper aims to develop a distributed least square algorithms over a network for colossal amounts of data. By doing so, we demonstrate the non-recursive least squares and the recursive forms of least squares estimation. In a distributed system, each agent of the system is connected through a communication network. In the network, each agent exchanges their information with neighbouring nodes, and all agents can collaboratively solve the same least square solution. However, the result of distributed least square estimation is valid for exponential convergence when we established a necessary and sufficient condition (3.3.9) on the step-size.

5 Matlab Code

AR(2) model:

X=1.2,3.2,2.4,2.2,2.3;

$a = 3.2^2 + 2.4^2 + 2.2^2$;

a

b=1.2*3.2+2.4*3.2+2.2*2.4+2.3*2.2;

b

$c = 1.2^2 + 3.2^2 + 2.4^2$;

c

xtx=[a b; b c];

xtx

d= -(2.4*3.2+2.2*2.4);

e= -(1.2*2.4+3.2*2.2+2.4*2.3);

xty=[d;e];

xty

theta=inv(xtx)*xty;

theta

Recursive least square example:

x = [30; 20];

```
xhat = [27; 19];
P = eye(2);
R = 0.02;
for k = 1 : 6
H = [k 0.79*(k-1)];
y = H * x + sqrt(R) * randn;
K = P * H' * inv(H * P * H' + R);
K
xhat = xhat + K * (y - H * xhat);
P = P - K * H * P;
P
end
x
xhat
```

OutPut:

```
K =
0.9804
0
P =
0.0196 0
0 1.0000
```

K =

0.0543

1.0934

P=

0.0175 -0.0429

-0.0429 0.1362

K =

-0.1380

0.7808

P=

0.0154 -0.0309

-0.0309 0.0686

K =

-0.1823

0.5995

P =

0.0132 -0.0238

-0.0238 0.0453 K =

-0.1879

0.4854

P=

0.0115 -0.0193

-0.0193 0.0337

x =

30

20

xhat =

30.0270

19.9705

Example of Distributed least square

I=eye(6);

I

H=[0 1; 2 0; 3 0; 2 0; 4 0; 1 0];

z= [-1; 2; 4; 1; 4; 5];

J=eye(2);

W= [0.4 0.2 0.4 0 0 0;

0.2 0.2 0.3 0.3 0 0;

0.4 0.3 0.15 0 0.15 0;

0 0.3 0 0.4 0.3 0;

0 0 0.15 0.3 0.3 0.25;

0 0 0 0 0.25 0.75]

$A = (I + W)^{(} - 2)$;

B=kron(A,J);

diatilH = [0 1 4 0 9 0 4 0 16 0 1 0];

```
tilH = diag(diatilH);
tilH
C=B*tilH;
C
e=eig(C);
e
e =
0
17.6865
8.4960
3.7131
1.4718
0.3187
0
0
0
0
0.7147
H=[0 1; 2 0; 3 0; 2 0; 4 0; 1 0];
z= [-1; 2; 4; 1; 4; 5];
theta = inv(transpose(H)*H) *transpose(H)*z ;
theta
```

```
W= [0.4 0.2 0.4 0 0 0;
0.2 0.2 0.3 0.3 0 0;
0.4 0.3 0.15 0 0.15 0;
0 0.3 0 0.4 0.3 0;
0 0 0.15 0.3 0.3 0.25;
0 0 0 0 0.25 0.75]
I =eye(2);
alpha = 0.0280;
x0 = [1; 5; 4; -5; 1; 5; 4; 2; 4; 9; 1; 3];
diatilH = [0 1 4 0 9 0 4 0 16 0 1 0];
tilH = diag(diatilH);
tilH
zh = [0;-1;4;0;12;0;2;0;16;0;5;0];
v0 = tilH*x0-zh;
v0
x1= kron(W,I)*x0-alpha*v0;
x1
v1 = kron(W,I)*v0+ tilH*x1-tilH*x0;
x2= kron(W,I)*x1-alpha*v1;
x2
v2= kron(W,I)*v1+ tilH*x2-tilH*x1;
[x, v] = deal(cell(3000, 1));
x1 = kron(W,I)*x0-alpha*v0 ;
```

```
v1 = kron(W,I)*v0+ tilH*x1-tilH*x0;
for k = 2:3000
xk = kron(W,I)*xk-1-alpha*vk-1 ;
vk= kron(W,I)*vk-1+ tilH*xk-tilH*xk-1 ;
end
celldisp(x)
y= [1:3000]
plot(y,[ x :   ])
xlabel(“time step”)
ylabel(“$x_i$”)
```

6 Reference

[1] A. K. Das and M. Mesbahi, "Distributed Linear Parameter Estimation over Wireless Sensor Networks," in IEEE Transactions on Aerospace and Electronic Systems, vol. 45, no. 4, pp. 1293-1306, Oct. 2009.

[2] D. Simon. Optimal State Estimations. John Wiley and Sons, Inc., Hoboken, New Jersey, 2006.

[3]Yang, T., George, J., Qin, J., Yi, X., and amp; Wu, J. (2020, January 05). Distributed least squares solver for network linear equations. Retrieved from https://arxiv.org/pdf/1810.00156.pdf

[4] Shumway, R. H., Stoffer, D. S. (2017). Time series analysis and its applications: with R examples. New York: Springer.

[5] Lai, T.L. and Robbins, H. (1977). Strong consistency of least-squares estimates in regression models. Proc. Natl. Acad. Sci. USA 74, 2667-2669.

[6] Models, A., Lai, T.L.,Ying, Z. (2006). EFFICIENT RECURSIVE ESTIMATION AND ADAPTIVE CONTROL IN STOCHASTIC REGRESSION AND ARMAX MODELS.

[7] Draper, N. R., & amp; Smith, H. (1998). Applied regression analysis. New York: John Wiley & amp; Sons.

[8] Madsen, H. (2008). The LS estimator for linear dynamic models. In Time series analysis. Boca Raton: Chapman & Hall/CRC.