Chapter 1 A Brief Review of Supervised Learning

There are a number of algorithms that are typically used for system identification, adaptive control, adaptive signal processing, and machine learning. These algorithms all have particular similarities and differences. However, they all need to process some type of experimental data. How we collect the data and process it determines the most suitable algorithm to use. In adaptive control, there is a device referred to as the *self-tuning regulator*. In this case, the algorithm measures the states as outputs, estimates the model parameters, and outputs the control signals. In reinforcement learning, the algorithms process rewards, estimates value functions, and output actions. Although one may refer to the recursive least squares (RLS) algorithm in the self-tuning regulator as a supervised learning algorithm and reinforcement learning as an unsupervised learning algorithm, they are both very similar.

1.1 Least Squares Estimates

The least squares (LS) algorithm is a well-known and robust algorithm for fitting experimental data to a model. The first step is for the user to define a mathematical structure or model that he/she believes will fit the data. The second step is to design an experiment to collect data under suitable conditions. “Suitable conditions” usually means the operating conditions under which the system will typically operate. The next step is to run the estimation algorithm, which can take several forms, and, finally, validate the identified or “learned” model. The LS algorithm is often used to fit the data. Let us look at the case of the classical two-dimensional linear regression fit that we are all familiar with:

$$y(n) = ax(n) + b \qquad (0)$$

$y(n)$

In this a simple linear regression model, where the input is the sampled signal and the output is . The model structure defined is a straight line. Therefore, we are assuming that the data collected will fit a straight line. This can be written in the form:

$$y(n) = \phi^T \theta \qquad (0)$$

where and . How one chooses determines the model structure, and this reflects how one believes the data should behave. This is the essence of machine learning, and virtually all university students will at some point learn the basic statistics of linear regression. Behind the computations of the linear regression algorithm is the scalar cost

$\phi^T = [a \quad b] \quad 1]$
function, given by

$$V = \sum_{n=1}^{N} (y(n) - \phi^T(n)\hat{\theta})^2$$

(0)

The term is the estimate of the LS parameter . The goal is for the estimate to minimize the cost function . To find the "optimal" value of the parameter estimate , one takes the partial derivative of the cost function with respect to and sets this derivative to zero. Therefore, one gets

(1)

$$\begin{aligned}\frac{\partial V}{\partial \hat{\theta}} &= \sum_{n=1}^{N} (y(n) - \phi^T(n)\hat{\theta})\phi(n) \\ &= \sum_{n=1}^{N} \phi(n)y(n) - \sum_{n=1}^{N} \phi(n)\phi^T(n)\hat{\theta}\end{aligned} \quad (1)$$

$$\frac{\partial V}{\partial \hat{\theta}} = 0$$

Setting , we get

$$\sum_{n=1}^{N} \phi(n)\phi^T(n)\hat{\theta} = \sum_{n=1}^{N} \phi(n)y(n)$$

(1)

$\hat{\theta}$
Solving for , we get the LS solution

$$\hat{\theta} = \left[\sum_{n=1}^{N} \phi(n)\phi^T(n)\right]^{-1} \left[\sum_{n=1}^{N} \phi(n)y(n)\right]$$

(1)

where the inverse, , exists. If the inverse does not exists, then the system is not identifiable. For example, if in the straight line case one only had a single point, then the inverse would not span the two-dimensional space and it would not exist. One needs at least two independent points to draw a straight line. Or, for example, if one had exactly the same point over and over again, then the inverse would not exist. One needs at least two independent points to draw a straight line. The matrix is referred to as the *information matrix* and is related to how well one can estimate the parameters. The inverse of the information matrix is the covariance matrix, and it is proportional to the variance of the parameter estimates. Both these matrices are positive definite and symmetric. These are very important properties which are used extensively in analyzing

$$P = \left[\sum_{n=1}^{N} \phi(n)\phi^T(n) \right]^{-1}$$

the behavior of the algorithm. In the literature, one will often see the covariance matrix referred to as . We can write the second equation on the right of Eq. in the form :

$$\frac{\partial V}{\partial \hat{\theta}} = 0 = \sum_{n=1}^{N} (y(n) - \phi^T(n)\hat{\theta})\phi(n) \tag{1}$$

and one can define the prediction errors as

$$\epsilon(n) = (y(n) - \phi^T(n)\hat{\theta}) \tag{1}$$

$\phi(n)$

The term within brackets in Eq. is known as the *prediction error* or, as some people will refer to it, the *innovations*. The term represents the error in predicting the output of the system. In this case, the output term is the correct answer, which is what we want to estimate. Since we know the correct answer, this is referred to as *supervised learning*. Notice that the value of the prediction error times the data vector is equal to zero. We then say that the prediction errors are orthogonal to the data, or that the data sits in the null space of the prediction errors. In simplistic terms, this means that, if one has chosen a good model structure , then the prediction errors should appear as white noise. Always plot the prediction errors as a quick check to see how good your predictor is. If the errors appear to be correlated (i.e., not white noise), then you can improve your model and get a better prediction.

One does not typically write the linear regression in the form of Eq. , but typically will add a white noise term, and then the linear regression takes the form

$$y(n) = \phi^T(n)\theta + \nu(n) \tag{1}$$

$\nu(n)$

where is a white noise term. Equation can represent an infinite number of possible model structures. For example, let us assume that we want to learn the dynamics of a second-order linear system or the parameters of a second-order infinite impulse response (IIR) filter. Then we could choose the second-order model structure given by

$$y(n) = -a_1 y(n-1) - a_2 y(n-2) + b_1 u(n-1) + b_2 u(n-2) + \nu(n) \tag{1}$$

$\phi(n)$

Then the model structure would be defined in as

$$\phi^T(n) = \begin{bmatrix} y(n-1) & y(n-2) & u(n-1) & u(n-2) \end{bmatrix} \tag{1}$$

In general, one can write an arbitrary th-order autoregressive exogenous (ARX) model structure as

k

$$\begin{aligned} y(n) = & -a_1 y(n-1) - a_2 y(n-2) - \ldots - a_m y(n-k) \\ & + b_1 u(n-1) + b_2 u(n-2) + .. + b_{n-k} u(n-k) + \nu(n) \end{aligned}^{(1)} \tag{1}$$

$\phi(n)$

and takes the form

$$\phi^T(n) = \begin{bmatrix} y(n-1)...y(n-m) & u(n-1)...u(n-m) \end{bmatrix} \quad (1)$$

$\phi(n)$

One then collects the data from a suitable experiment (easier said than done!), and then computes the parameters using Eq. The vector can take many different forms; in fact, it can contain nonlinear functions of the data, for example, logarithmic terms or square terms, and it can have different delay terms. To a large degree, one can use ones professional judgment as to what to put into . One will often write the data in the matrix form, in which case the matrix is defined as

$$\mathbf{\Phi} = \begin{bmatrix} \phi(1) & \phi(2) & ... & \phi(N) \end{bmatrix} \quad (1)$$

and the output matrix as

$$\mathbf{Y} = \begin{bmatrix} y(1) & y(2) & ... & y(N) \end{bmatrix} \quad (1)$$

Then one can write the LS estimate as

$$\hat{\Theta} = (\mathbf{\Phi}\mathbf{\Phi}^T)^{-1}\mathbf{\Phi}\mathbf{Y} \quad (1)$$

Furthermore, one can write the prediction errors as

$$\mathbf{E} = \mathbf{Y} - \mathbf{\Phi}^T\hat{\Theta} \quad (2)$$

We can also write the orthogonal condition as $\mathbf{\Phi}E = 0$

The LS method of parameter identification or machine learning is very well developed and there are many properties associated with the technique. In fact, much of the work in statistical inference is derived from the few equations described in this section. This is the beginning of many scientific investigations including work in the social sciences.