Lecture 12 - Notes

Rahul Arya

February 2019

1 Overview

From the past few lectures, we now know, given a discrete-time linear system, how to apply inputs to drive it towards a desired state, even when our inputs were transformed by a matrix B. However, in doing so, we relied on a complete knowledge of the initial state $\vec{x}[0]$. Now, we will consider the dual problem of *observability* - given limited observations of the system's evolution over time, can we determine its initial internal state $\vec{x}[0]$?

2 The Observability Problem

First, we will state the problem more explicitly. Consider the following discrete system:

$$\begin{aligned}\vec{x}[i+1] &= A\vec{x}[i] + B\vec{u}[i]\\ \vec{y}[i] &= C\vec{x}[i].\end{aligned}$$

The system will evolve over time due to a series of control inputs $\vec{u}[i]$ that we apply, in the following manner:

$$\begin{aligned}\vec{x}[0] &= \vec{x}[0]\\ \vec{x}[1] &= A\vec{x}[0] + B\vec{u}[0]\\ \vec{x}[2] &= A^2\vec{x}[0] + AB\vec{u}[0] + B\vec{u}[1]\\ &\vdots\end{aligned}$$

Clearly, if we could see the states $\vec{x}[0], \vec{x}[1], \vec{x}[2], \ldots$ it would be trivial to recover $\vec{x}[0]$ - after all, we are told it explicitly! However, in physical systems it is very rare for us to be able to view the entirety of the state at any given time. Recall that we can use these discrete time models to approximate the behavior of an analog circuit with varying input. In such a scenario, it would be very unlikely for us to be able to measure *every* current, voltage, and charge throughout the system.

Instead, we may only be able to observe a linearly transformed version of the state, which we denote as $\vec{y}$. Thus, the question becomes - given the sequence of observations

$$(\vec{y}[0], \vec{u}[0]), (\vec{y}[1], \vec{u}[1]), (\vec{y}[2], \vec{u}[2]), \ldots,$$

can we recover the initial state $\vec{x}[0]$?

3 Simplifying the Problem

To do so, we will first consider a simpler problem, when no input is present in the system. In other words, the system can be thought of as:

$$\begin{aligned} \vec{x}[i+1] &= A\vec{x}[i] \\ \vec{y}[i] &= C\vec{x}[i]. \end{aligned}$$

This simplification is convenient since it means that we no longer have to worry about choosing our inputs. From this, we can immediately express

$$\vec{y}[0] = C\vec{x}[0].$$

If C were invertible, then it would be trivial to recover $\vec{x}[0]$ without even letting the system evolve. However, typically C is a "wide" matrix, taking in a higher dimensional state vector and returning a low dimensional vector of observations, so that tends not to be the case. We use n to denote the dimension of our state vector $\vec{x}$, and k to denote the dimension of the observation vector $\vec{y}$, so C has k rows and n columns.

However, we saw last time when we looked at observability that allowing a system more time to evolve gave us more control over its behavior. It is natural to conjecture that allowing our system more time to evolve will similarly allow us to obtain more information about its state. Indeed, after i time steps, we will obtain the equations

$$\begin{aligned} \vec{y}[0] &= C\vec{x}[0] \\ \vec{y}[1] &= CA\vec{x}[0] \\ \vec{y}[2] &= CA^2\vec{x}[0] \\ &\vdots \\ \vec{y}[i] &= CA^i\vec{x}[0]. \end{aligned}$$

Rewriting them in stacked matrix form, we obtain the system

$$\begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^i \end{bmatrix} \vec{x}[0] = \begin{bmatrix} \vec{y}[0] \\ \vec{y}[1] \\ \vec{y}[2] \\ \vdots \\ \vec{y}[i] \end{bmatrix}.$$

We typically refer to the matrix on the left as the observability matrix, denoted as $\mathscr{O}$. Therefore, we can use least squares to recover $\vec{x}[0]$, to obtain

$$\vec{x}[0] = (\mathscr{O}^T\mathscr{O})^{-1}\mathscr{O}^T \begin{bmatrix} \vec{y}[0] \\ \vec{y}[1] \\ \vec{y}[2] \\ \vdots \\ \vec{y}[i] \end{bmatrix}.$$

From EE16A, we know that this approach works exactly when $\mathscr{O}$ does not have a nontrivial nullspace (i.e. has linearly independent columns).

4 Observability with Inputs

Now, we will return to the general case, when we can supply inputs $\vec{u}[i]$. Clearly,

$$\begin{aligned}
\vec{y}[0] &= C\vec{x}[0] \\
\vec{y}[1] &= CA\vec{x}[0] + CB\vec{u}[0] \\
\vec{y}[2] &= CA^2\vec{x}[0] + CAB\vec{u}[0] + CB\vec{u}[1] \\
&\vdots \\
\vec{y}[i] &= CA^i\vec{x}[0] + CA^{i-1}B\vec{u}[0] + CA^{i-2}B\vec{u}[1] + \ldots + CB\vec{u}[i-1].
\end{aligned}$$

In addition, let us define $\vec{y}_{free}$ to be the observations that we would have gotten had our inputs all been 0. From the previous section, we know that

$$\begin{aligned}
\vec{y}_{free}[0] &= C\vec{x}[0] \\
\vec{y}_{free}[1] &= CA\vec{x}[0] \\
\vec{y}_{free}[2] &= CA^2\vec{x}[0] \\
&\vdots \\
\vec{y}_{free}[i] &= CA^i\vec{x}[0].
\end{aligned}$$

Observe that we may rewrite any given $\vec{y}[j]$ as follows

$$\begin{aligned}
\vec{y}[j] &= CA^j\vec{x}[0] + CA^{j-1}B\vec{u}[0] + CA^{j-2}B\vec{u}[1] + \ldots + CB\vec{u}[j-1] \\
&= CA^j\vec{x}[0] + \sum_{k=0}^{j-1} CA^{j-k-1}B\vec{u}[k] \\
&= \vec{y}_{free}[j] + \sum_{k=0}^{j-1} CA^{j-k-1}B\vec{u}[k].
\end{aligned}$$

Critically, notice that the second term in the sum for $\vec{y}[j]$ has no dependence on the initial state, and so can be determined exactly given only the system and the applied inputs. Therefore, it gives us neither more or less information about the initial state, no matter what inputs we apply. As a result, we can focus on the case when $\vec{u} = 0$, since we will always be able to recover $\vec{y}_{free}$ from $\vec{y}$ by subtracting out the known dependence on the inputs.

Recall from last lecture that a system is controllable in infinite timesteps if and only if it is controllable in at most n timesteps. The exact same proof applies here in the context of observability, and so will not be repeated. That is to say, we can determine the initial state $\vec{x}[0]$ after an arbitrary many number of observations made each timestep if and only if we can do so in at most n timesteps. Therefore, we typically use $\mathscr{O}$ to refer to the observations over n timesteps, so we have

$$\mathscr{O} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}.$$

Consequently, when attempting to determine the initial state of a system, we should apply arbitrary inputs (or have them applied for us, if we cannot control the inputs), observe the first n values of $\vec{y}[i]$ (from $\vec{y}[0]$ through $\vec{y}[n-1]$), use our knowledge of the inputs to deduce the corresponding $\vec{y}_{free}$, and solve the least squares system for $\vec{x}[0]$ (or report that no unique solution exists).

5 Some Examples

To complete our discussion of observability, we will look at some toy examples. Consider

$$A = \begin{bmatrix} 1 & -1 \\ 2 & 1 \end{bmatrix}$$
$$C = \begin{bmatrix} 1 & 0 \end{bmatrix}.$$

In other words, we can only observe the first component of our state vector at any given time. Notice that since our inputs are of no importance is determining the observability of a system, we do not need to know B.

We see that the observability matrix is

$$\mathscr{O} = \begin{bmatrix} 1 & 0 \\ 1 & -1 \end{bmatrix}.$$

Clearly, $\mathscr{O}$ is of full rank, so we should be able to recover the initial state $\vec{x}[0]$ from the first $n = 2$ observations. Specifically, if $\vec{x}[0] = \begin{bmatrix} a \\ b \end{bmatrix}$, we obtain

$$\begin{aligned} \vec{x}[1] &= \begin{bmatrix} a - b \\ 2a + b \end{bmatrix} \\ \implies \quad \vec{y}[0] &= \begin{bmatrix} a \end{bmatrix} \\ \vec{y}[1] &= \begin{bmatrix} a - b \end{bmatrix}. \end{aligned}$$

Clearly, given a and $a - b$, we can recover both components of the original state vector exactly.

However, now consider the alternative system

$$\begin{aligned} A &= \begin{bmatrix} 2 & 0 \\ 2 & 3 \end{bmatrix} \\ C &= \begin{bmatrix} 1 & 0 \end{bmatrix}. \end{aligned}$$

We see that the observability matrix is

$$\mathscr{O} = \begin{bmatrix} 1 & 0 \\ 2 & 0 \end{bmatrix},$$

so we will not be able to recover the initial state.

6 System Identification

We now know how, given A, B, and a sequence of control inputs and observed responses $(\vec{u}, \vec{y})$, how to deduce the initial state $\vec{x}[0]$. We will now go one step further - given only $\vec{u}$ and $\vec{y}$, we will attempt to determine A and B, in order to identify the characteristics of our system. This process is of great importance, as when constructing real-world mechanical systems, it is practically impossible to determine all their behaviors theoretically - we must do so empirically.

We will first consider a simpler version of the problem, where C is known to be the identity matrix, and no noise is present. As we have seen many times, since $C = I$, we have that

$$\vec{y}[i + 1] = A\vec{x}[i] + B\vec{u}[i].$$

Let's also express our unknown matrices A and B in terms of scalars, as follows:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$$
$$B = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1k} \\ b_{21} & b_{22} & \dots & b_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nk} \end{bmatrix}.$$

Substituting into our relation for $\vec{y}$, we obtain

$$\begin{bmatrix} y_1[i+1] \\ y_2[i+1] \\ \\ y_n[i+1] \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1[i] \\ x_2[i] \\ \vdots \\ x_n[i] \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1k} \\ b_{21} & b_{22} & \dots & b_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nk} \end{bmatrix} \begin{bmatrix} u_1[i] \\ u_2[i] \\ \vdots \\ u_k[i] \end{bmatrix},$$

breaking down our state, observation, and input vectors into their scalar components as well.

Now comes the interesting part. Recall that our unknowns are in fact all the a_{ij} and b_{ij}, and our knowns are all the $\vec{y}_i$, $\vec{x}_i$, and $\vec{u}_i$. When solving linear systems, we know that we should put our unknowns into a vector. By considering each row of the above matrix separately, we obtain scalar equations of the form

$$y_j[i+1] = \begin{bmatrix} a_{j1} & a_{j2} & \dots & a_{jn} \end{bmatrix} \begin{bmatrix} x_1[i] \\ x_2[i] \\ \vdots \\ x_n[i] \end{bmatrix} + \begin{bmatrix} b_{j1} & b_{j2} & \dots & b_{jk} \end{bmatrix} \begin{bmatrix} u_1[i] \\ u_2[i] \\ \vdots \\ u_k[i] \end{bmatrix}$$

for all $1 \leq j \leq n$. Notice that the two terms on the right of the equation are really just dot products, producing a scalar. As the dot product is symmetric in $\mathbb{R}$, we may rewrite the above equation as

$$y_j[i+1] = \begin{bmatrix} x_1[i] & x_2[i] & \cdots & x_n[i] \end{bmatrix} \begin{bmatrix} a_{j1} \\ a_{j2} \\ \vdots \\ a_{jn} \end{bmatrix} + \begin{bmatrix} u_1[i] & u_2[i] & \cdots & u_k[i] \end{bmatrix} \begin{bmatrix} b_{j1} \\ b_{j2} \\ \vdots \\ b_{jk} \end{bmatrix}$$

again for all $1 \leq j \leq n$.

Combining the two terms on the right, we finally obtain the equation

$$y_j[i+1] = \begin{bmatrix} x_1[i] & x_2[i] & \cdots & x_n[i] & u_1[i] & u_2[i] & \cdots & u_k[i] \end{bmatrix} \begin{bmatrix} a_{j1} \\ a_{j2} \\ \vdots \\ a_{jn} \\ b_{j1} \\ b_{j2} \\ \vdots \\ b_{jk} \end{bmatrix}$$

again for all $1 \leq j \leq n$.

Notice that we have managed to place all the unknowns involving the jth element of the state in a single vector. Since we now have a linear equation with our unknowns in a vector, are we done, since we can do Gaussian elimination to solve for the unknowns?

Unfortunately not. Notice that there are $n + k$ unknowns, but only a single equation. We could try to get more equations by considering all values of j simultaneously, but that would also bring in new unknowns, since we'd have to consider the a_{ji} and b_{ji} for *all* j. So is all hope lost?

No! Recall that our equation holds for *all* values of i, since we are assuming that our system's transition equation does not vary over time! Notice that for all timesteps $0 < i \leq t$, the vector of unknowns remains the same. Thus, by considering all these values of i and stacking them vertically, we obtain the system

$$\begin{bmatrix} y_j[1] \\ y_j[2] \\ \vdots \\ y_j[t] \end{bmatrix} = \begin{bmatrix} x_1[0] & \cdots & x_n[0] & u_1[0] & \cdots & u_k[0] \\ x_1[1] & \cdots & x_n[1] & u_1[1] & \cdots & u_k[1] \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ x_1[t-1] & \cdots & x_n[t-1] & u_1[t-1] & \cdots & u_k[t-1] \end{bmatrix} \begin{bmatrix} a_{j1} \\ a_{j2} \\ \vdots \\ a_{jn} \\ b_{j1} \\ b_{j2} \\ \vdots \\ b_{jk} \end{bmatrix}$$

which consists of t equations, not just 1! For $t \geq n+j$, we can use least squares to solve for our unknowns, and repeat the process for each value of j in order to fully determine our system. Typically, we create a matrix P of our unknowns across all values of j by stacking them horizontally to obtain

$$P = \begin{bmatrix} a_{11} & a_{21} & \dots & a_{n1} \\ a_{12} & a_{22} & \dots & a_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \dots & a_{nn} \\ b_{11} & b_{21} & \dots & b_{n1} \\ b_{12} & b_{22} & \dots & b_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ b_{1k} & b_{2k} & \dots & b_{nk} \end{bmatrix} = \begin{bmatrix} A^T \\ B^T \end{bmatrix}.$$

We can follow a similar stacking procedure to obtain

$$D = \begin{bmatrix} x_1[0] & \cdots & x_n[0] & u_1[0] & \cdots & u_k[0] & \\ x_1[1] & \cdots & x_n[1] & u_1[1] & \cdots & u_k[1] & \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \\ x_1[t-1] & \cdots & x_n[t-1] & u_1[t-1] & \cdots & u_k[t-1] & \end{bmatrix}$$

and

$$S = \begin{bmatrix} y_1[1] & y_2[1] & \cdots & y_n[1] \\ y_1[2] & y_2[2] & \cdots & y_n[2] \\ \vdots & \ddots & \vdots & \\ y_1[t] & y_2[t] & \cdots & y_n[t] \end{bmatrix}.$$

Thus, we can combine all our values of j into the single equation

$$DP = S$$

and solve using least squares to obtain

$$P = (D^T D)^{-1} D^T S.$$

Recall that $D^T D$ is invertible exactly when D has linearly independent columns. We will discuss what happens when the columns made out of the x_j are dependent later - for now, we will simply comment that choosing our inputs randomly will ensure with high probability that the input columns are all linearly independent both of each other and of the earlier x_j columns.

7 Scalar-Output System Identification

However, all of these calculations have assumed that we have perfect knowledge of the state $\vec{x}$ at every timestep. But as we saw when considering the problem of observability, such a scenario is very unlikely. In this section, we will demonstrate that receiving even just a one-dimensional output scalar y can be sufficient to fully determine the behavior of a system.

We can imagine such a system as follows:

$$\begin{tikzcd}\text{u[i]} \arrow[r] & \boxed{\text{Unknown System}} \arrow[r] & \text{y[i]}\end{tikzcd}$$

Every timestep, we supply an input scalar $u[i]$, and observe a scalar output $y[i]$.

To model such a system, we will make the following observation. For the purposes of system identification, we *don't* actually care about the matrices A and B. What we are interested in is in the modelling of our system's *behavior*. Specifically, given sufficiently many past $(\vec{u}, \vec{y})$ pairs, we want to predict our observations of the state in response to arbitrary future inputs. Thus, it doesn't actually matter whether our model's internal state is the same as that of the true system, so long as it behaves identically for all possible inputs.

Consider the following model of a system with scalar input $\vec{u}$ and scalar output y:

$$y[i+1] = a_{11}y[i] + \ldots + a_{n1}y[i-n+1] + b_1u[i] + \ldots + b_n[i-n+1].$$

It can be proved, though we will not do so here, that *any* observable system with an n-dimensional internal state can be modelled as the above, in what is known as *observable canonical form*.

Instead, we will justify this claim intuitively. If a system is observable, then (given A and B) we can deduce its internal state from n consecutive observations of both u and y. From its internal state, we can then apply the state transition equation to determine the next output $y[i+1]$. Since our model has access to the last n values of both u and y, and since everything is linear, it should make sense that $y[i+1]$ can be represented as a linear combination of the past n values of u and y, which is what we do here.

If we accept that our system can be modelled in this fashion, then a straightforward least squares computation can be performed to deduce the coefficients a_{i1} and b_i. Specifically, we can rearrange and stack our state equation and combine additive terms, just as we did in the previous section, to obtain

$$\begin{bmatrix} y[n] \\ y[n+1] \\ \vdots \\ y[n+t] \end{bmatrix} = \begin{bmatrix} y[n-1] & \cdots & y[0] & u[n-1] & \cdots & u[0] \\ y[n] & \cdots & y[1] & u[n] & \cdots & u[1] \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ y[n+t-1] & \cdots & y[t] & u[n+t-1] & \cdots & u[t] \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{n1} \\ b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

We will establish some terminology analogous to those that we saw previously, such that

$$\begin{aligned}\vec{p} &= \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{n1} & b_1 & b_2 & \cdots & b_n \end{bmatrix}^T \\ \vec{s} &= \begin{bmatrix} y[n] & y[n+1] & \cdots & y[n+t] \end{bmatrix}^T \\ D &= \begin{bmatrix} y[n-1] & \cdots & y[0] & u[n-1] & \cdots & u[0] \\ y[n] & \cdots & y[1] & u[n] & \cdots & u[1] \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ y[n+t-1] & \cdots & y[t] & u[n+t-1] & \cdots & u[t] \end{bmatrix},\end{aligned}$$

so we can rewrite our equation as

$$D\vec{p} = \vec{s}.$$

Finally, by least squares, we obtain the solution

$$\vec{p} = (D^T D)^{-1} D^T \vec{s}.$$

Of course, the only remaining issue is in ensuring that $D^T D$ is in fact invertible. We have already mentioned that we must choose inputs that form linearly independent columns, and that one good way to do this is to choose inputs randomly. However, it is also possible that the y columns of D are linearly independent. Imagine, for instance, a system with no memory, so $y[n+1] = x[n+1] = u[n]$. Then clearly the y columns would be linearly dependent with the u columns, no matter how we chose our inputs.

While we will not prove this formally, such linear dependence is ultimately caused by a poor choice of n. Recall that we just assumed the dimension of our system's internal state vector to be n. If this choice is too low, we will discover that quickly, as our predictions fail to match reality. It should be at least intuitively clear that if we choose to high a value of n, we will in some sense "overfit" our observations, producing linearly dependent columns and failing to make accurate predictions, meaning that the choice of n is important when performing this sort of system identification.