

9 Building generative models: structural learning, and identification of the learner

Suppose that it is true that when the brain generates a motor command, it also predicts the sensory consequences. The state estimation framework that we developed in the last few chapters suggests that when the actual sensory measurements arrive, the brain should combine what it predicted with what it measured to form an estimate of the state of the body (and whatever else contributes to things that it can sense). By doing so, it can estimate this state better than if it were to rely on the sensors alone. For example, we can estimate the location of a visual stimulus better if we can predict its position as well as see it, as compared to if we can only see it but not predict it (Vaziri et al., 2006). However, this last statement is true only if the predictions that the brain makes are unbiased.

Being an unbiased predictor is actually rather difficult, because our body and the objects we interact with have dynamics: inputs (motor commands) and outputs (sensory measurements) are not related via some static mapping. Rather, dynamics implies that state of the system changes both as a function of time (passive dynamics), and as a function of the input (active dynamics). What's more, the relationship between inputs and outputs can change (for example, muscles can fatigue), making it necessary for us to adjust our predictions. We saw that one way our brain can learn to make better predictions is to assume a generative model that describes the relationship between inputs and outputs via some hidden states (for example, the fatigue state of the muscle can be one of these states). The prediction errors can guide the process of estimating these hidden states, and thereby improving the predictions.

The elephant in the room is the generative model itself. Where does the brain get such a model? That is, how does our brain go about finding (presumably, learning) a model that has a *structure* or topology that can in principle represent the relationship between inputs (e.g., motor commands) and observations (sensory measurements)? To accurately predict the sensory consequences of a motor command we need to be able to have a model that can approximate behavior of a dynamical system. We are no longer talking about the hidden states of some generic model, but rather the topology of the model itself. How do we discover this topology, or structure of the dynamical system that we wish to control? The problem that our brain faces is one of system identification.

Consider the problem of learning to control motion of a ping pong paddle. It would be useful if during practice, we learned a model that had a structure with hidden states that could also help us with learning control of a tennis racket. This should be possible because the two objects are rigid-body inertial systems with dynamics that are similar in structure. If we could somehow discover this structure during playing ping pong, it can vastly speed up our learning of tennis. In this chapter, we will consider this problem of structural learning.

How is structural learning different than the learning that we had considered in the last few chapters? The problem of learning an internal model, a model that can accurately predict sensory consequence of movements, can be approached from two perspectives. In the perspective that we had considered in the last few chapters (which we called the state-estimation perspective), we started with a specific model that could in principle represent the relationship between inputs and outputs, and then estimated the hidden states of this model based on our prediction errors. We did not consider where such a model with these specific hidden states might come from. In the perspective that we will consider here, the problem is to discover this structure from the relationships (input – outputs) that we observe. Structural learning relies on the long-term history of the motor commands and their sensory consequences.

9.1 Structure of dynamics for two example systems

Suppose that you want to build a model that can help you predict motion of your arm. Because you tend to hold different kinds of objects in your hand (and these objects change the arm's dynamics), you want this model to be flexible enough so that it can readily predict motion in different scenarios. Let us sketch what this model might look like.

For simplicity, let us assume that the arm will move in the horizontal plane. The relationship between torques at the shoulder τ_1 and elbow τ_2 and motion (joint angular position q_1 and q_2 , and their derivatives) for the system shown in Fig. 9.1A is:

$$\begin{aligned} \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} &= \begin{bmatrix} a_1 + 2a_3 \cos(q_2) + a_2 & a_2 + a_3 \cos(q_2) \\ a_2 + a_3 \cos(q_2) & a_2 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} \\ &+ \begin{bmatrix} -a_3 \dot{q}_2 \sin(q_2) & -a_3 (\dot{q}_1 + \dot{q}_2) \sin(q_2) \\ a_3 \sin(q_2) \dot{q}_1 & 0 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} \end{aligned} \quad (9.1)$$

[If you are interested in how to derive these equations, check out the Introduction to Dynamics lecture notes at: <http://www.shadmehrlab.org/book/dynamics.pdf>]. Eq. (9.1) is called *inverse dynamics* because it maps states to forces. Eq. (9.1) is basically the familiar $F = m\ddot{x}$ but written for the complicated system of Fig. 9.1, in coordinates of torques and joint rotations. *Forward dynamics* is the map from forces to change in states. In Eq. (9.1), the parameters a_i are constants that depend on the mass properties of the arm:

$$\begin{aligned} a_1 &= m_1 |x_1|^2 + I_1 + m_2 l_1^2 \\ a_2 &= m_2 |x_2|^2 + I_2 \\ a_3 &= m_2 l_1 |x_2| \end{aligned} \quad (9.2)$$

where m_1 and m_2 are masses of the upper arm and forearm, $|x_1|$ and $|x_2|$ are lengths of the vector that points to the center of the mass of each segment, I_1 and I_2 are the moments of inertia of each segment, and l_1 is the length of the upper arm. We can rewrite Eq. (9.1) using vector notation:

$$\boldsymbol{\tau} = H(\mathbf{a}, \mathbf{q})\ddot{\mathbf{q}} + \mathbf{c}(\mathbf{a}, \mathbf{q}, \dot{\mathbf{q}}) \quad (9.3)$$

Matrix H represents inertia of the arm. Inertia depends on parameter \mathbf{a} (some combination of mass and link lengths of the arm, as specified in Eq. 9.2), and position of the arm \mathbf{q} . Vector \mathbf{c} represents the centripetal and coriolis forces. These force also depend on parameter \mathbf{a} , as well as position and velocity of the arm. A fundamental characteristic of Eq. (9.3) is that the torques that are due to acceleration are linearly separable from forces that are due to velocity. A second important characteristic is that the parameter \mathbf{a} appears linearly, i.e., torques are a linear function of the parameter that might change if we were to hold different objects in our hand.

To see this last point, let us consider what happens when you hold an object in your hand. When you hold an object in your hand like a cup of coffee, the addition of this mass increases m_2 , I_2 , and $|x_2|$. This means that the equations of motion for your arm when you are holding a cup has the same structure as when you are not holding the cup, with the only difference being in the parameters a_i . That is, holding a cup changes \mathbf{a} , but nothing else.

What happens when you hold a tennis racket? Unlike the coffee cup, the racket shifts the center of mass off the forearm (Fig. 9.1B), resulting in the following equations of motion:

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} -h\dot{q}_2 & -h(\dot{q}_1 + \dot{q}_2) \\ h\dot{q}_1 & 0 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} \quad (9.4)$$

where

$$\begin{aligned} H_{11} &= a_1 + 2a_3 \cos(q_2) + 2a_4 \sin(q_2) \\ H_{12} &= H_{21} = a_2 + a_3 \cos(q_2) + a_4 \sin(q_2) \\ H_{22} &= a_2 \\ h &= a_3 \sin(q_2) - a_4 \cos(q_2) \end{aligned} \quad (9.5)$$

and

$$\begin{aligned} a_1 &= m_1 |x_1|^2 + I_1 + I_e + m_e |x_e|^2 + m_e l_1^2 \\ a_2 &= m_e |x_e|^2 + I_e \\ a_3 &= m_e l_1 |x_e| \cos(\phi) \\ a_4 &= m_e l_1 |x_e| \sin(\phi) \end{aligned} \quad (9.6)$$

Notice that switching from a cup (roughly a point mass) to a racket adds one parameter to our equation, but maintains the fact that forces that depend on acceleration and velocity remain separable. Now if we were to switch from holding a tennis racket to a ping pong paddle (or any other rigid object), once again all that changes are the parameters a_i , with no change in the structure of our relationship in Eq. (9.4). In sum, we see that physics implies regularity in the relationship between motion and forces. In particular, for our arm the forces that are produced due to acceleration and velocity are linearly separable. Furthermore, regardless of the rigid object that we may hold in our hand, the forces remain linear in terms of parameters representing masses, lengths, etc.

Now suppose that we knew this structure and wanted to use that information to estimate the parameters a_i . From Eq. (9.3), we have:

$$\begin{aligned} \ddot{\mathbf{q}} &= -H^{-1}(\mathbf{a}, \mathbf{q}) \mathbf{c}(\mathbf{a}, \mathbf{q}, \dot{\mathbf{q}}) + H^{-1}(\mathbf{a}, \mathbf{q}) \boldsymbol{\tau} \\ &= G(\mathbf{a}, \mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\tau}) \end{aligned} \quad (9.7)$$

Eq. (9.7) is parameterized by a small number of unknowns, the elements a_i of the vector \mathbf{a} .

These are some of the hidden states of the system, which we can estimate from sensory observations. To use our state estimation framework, we need to linearize the relationship between sensory observations and the states that we wish to estimate. Let us label the states of

our system as the column vector $\mathbf{x} = [\mathbf{q}, \dot{\mathbf{q}}, \mathbf{a}]$. Using Eq. (9.7), we can approximate $\ddot{\mathbf{q}}$ around our current estimate $\hat{\mathbf{x}}$:

$$\ddot{\mathbf{q}} = G(\hat{\mathbf{x}}, \boldsymbol{\tau}) + \left. \frac{dG}{d\mathbf{x}} \right|_{\hat{\mathbf{x}}} (\mathbf{x} - \hat{\mathbf{x}}) \quad (9.8)$$

and then write the state update equation as a linear function of current state $\mathbf{x}(t)$ and some constant terms:

$$\begin{aligned} \mathbf{q}(t + \Delta) &= \mathbf{q}(t) + \dot{\mathbf{q}}(t) \Delta \\ \dot{\mathbf{q}}(t + \Delta) &= \dot{\mathbf{q}}(t) + \left. \frac{dG}{d\mathbf{x}} \right|_{\hat{\mathbf{x}}} \mathbf{x}(t) \Delta - \left. \frac{dG}{d\mathbf{x}} \right|_{\hat{\mathbf{x}}} \hat{\mathbf{x}} \Delta + G(\hat{\mathbf{x}}, \boldsymbol{\tau}) \Delta \\ \mathbf{a}(t + \Delta) &= A\mathbf{a}(t) + \boldsymbol{\varepsilon}_a \end{aligned} \quad (9.9)$$

The terms on the second line of Eq. (9.9) are simply an expansion of velocity in terms of acceleration, i.e., $\dot{\mathbf{q}}(t + \Delta) = \dot{\mathbf{q}}(t) + \ddot{\mathbf{q}}(t) \Delta$. The last line in Eq. (9.9) is the state equation regarding the parameters \mathbf{a} , which includes state noise. The measurement equation is:

$$\mathbf{y}(t) = L\mathbf{x}(t) + \boldsymbol{\varepsilon}_y \quad (9.10)$$

In Eq. (9.10), the matrix L indicates the state variables that we can observe (typically position and velocity). Having approximated our non-linear system with a linear equation, we can now use state estimation techniques to form an estimate $\hat{\mathbf{x}}$, which includes the parameter that we are looking for $\hat{\mathbf{a}}$.

The point is that there exists a model that has the appropriate structure for the dynamics that we wish to approximate. In this model, the problem of representing dynamics of different objects that we might hold in our hand reduces to changes in the space spanned by the vector \mathbf{a} . Within this 4D space we can represent arm dynamics for a large range of objects.

9.2 Evidence for learning a structural model

Daniel Braun, Ad Aertsen, Daniel Wolpert, and Carsten Mehring (2009) argued that if during learning, people adjust not just the parameters of a default generative model, but learn a new one, then this new model should help them with learning of tasks that are also supported by the same structure (but perhaps with different parameter values). The idea is that if you learn to ride one kind of bike (say a mountain bike), you should be able to rapidly learn to ride a race bike. If you learn to play ping-pong, it should also help you learn tennis. To test for this structure specific

facilitation, they exposed a group of subjects to a long period of visuomotor adaptation during reaching. The caveat was that from trial-to-trial, the perturbations were uniformly distributed between -90° and $+90^\circ$. If learning was merely an updating of parameter values for an existing model, then random perturbations should produce no sustained change in these parameter values (as the mean perturbation is zero). After this period of random rotation perturbations, they presented subjects with a constant $+60^\circ$ rotation. They found that compared to naïve subjects, the people with the prior exposure to the random rotations were much better in learning the constant rotation (Fig. 2, random rotation group). Next, they considered prior training that had random errors, but not of the rotation class. In this group, the random errors were generated in a long set of trials in which the movements were transformed by a rotation, a shearing, and a scaling (a linear transformation). This group performed significantly worse than the group that had prior training in the rotation perturbation (Fig. 2, random linear transform group).

Therefore, the prior exposure to a rotation perturbation, despite being random and unlearnable, appeared to significantly improve learning rates for a member of the same perturbation class. This is consistent with the idea that during exposure to the rotation class of perturbations, people learned the structure of the perturbation, despite being unable to learn the specific parameter of that perturbation, and this structure aided their learning when they subsequently were exposed to a constant rotation.

9.3 Non-uniqueness of the structure

The problem that we have been discussing is closely related to the general system identification problem where inputs $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \dots$ are given to a system and outputs $\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \dots$ are measured and the intention is to identify the dynamics of that system. We do not know how to solve this problem in general, but we can solve it if we assume that the system that we are trying to identify has a linear structure. Our objective here is to show the computations that are an essential part of finding the correct structure.

We will assume that dynamics of the system that we are trying to model is of the form shown below, with hidden states \mathbf{x} that are of unknown dimensionality, and with unknown matrices A , B , C , and D :

$$\begin{aligned}\mathbf{x}^{(n+1)} &= A\mathbf{x}^{(n)} + B\mathbf{u}^{(n)} + \boldsymbol{\varepsilon}_x^{(n)} & \boldsymbol{\varepsilon}_x &\square N(0, Q) \\ \mathbf{y}^{(n)} &= C\mathbf{x}^{(n)} + D\mathbf{u}^{(n)} + \boldsymbol{\varepsilon}_y^{(n)} & \boldsymbol{\varepsilon}_y &\square N(0, R)\end{aligned}$$

The objective is to find the structure, i.e., matrices A , B , C , and D , and noise properties Q and R , that describes the relationship between the sequence of inputs $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \dots$ that we gave to the system and the sequence of measurements $\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \dots$ that we made from the system. We will identify the generative model that produced the data that we observed.

An elegant technique for identifying the system in Eq. (9.8) is via *subspace analysis* (van Overschee and De Moor, 1996). Let us begin with the deterministic problem in which our system is without noise. That is, let us suppose that the structure that we wish to identify has the following form:

$$\begin{aligned}\mathbf{x}^{(n+1)} &= A\mathbf{x}^{(n)} + B\mathbf{u}^{(n)} \\ \mathbf{y}^{(n)} &= C\mathbf{x}^{(n)} + D\mathbf{u}^{(n)}\end{aligned}\tag{9.11}$$

We will provide a sequence of inputs $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \dots$ to this system, and measure outputs $\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \dots$. The parameters that we are searching for are A , B , C , D , and $\mathbf{x}^{(1)}$ (the initial conditions). However, there are an infinite number of parameters that can give us this exact input-output sequence. That is, there is no unique solution to our problem. To see this, from Eq. (9.11) we can write:

$$\begin{bmatrix} \mathbf{x}^{(2)} & \mathbf{x}^{(3)} & \dots & \mathbf{x}^{(p+1)} \\ \mathbf{y}^{(1)} & \mathbf{y}^{(2)} & \dots & \mathbf{y}^{(p)} \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots & \mathbf{x}^{(p)} \\ \mathbf{u}^{(1)} & \mathbf{u}^{(2)} & \dots & \mathbf{u}^{(p)} \end{bmatrix}\tag{9.12}$$

In Eq. (9.12), we have arranged the vectors and matrices to form new matrices. For example, the matrix on the left side of Eq. (9.12) is composed of columns with elements that are specified by column vectors \mathbf{x} and \mathbf{y} . If we now multiply the state equation by an arbitrary but invertible matrix T , we have:

$$\begin{bmatrix} T & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(2)} & \mathbf{x}^{(3)} & \dots & \mathbf{x}^{(p+1)} \\ \mathbf{y}^{(1)} & \mathbf{y}^{(2)} & \dots & \mathbf{y}^{(p)} \end{bmatrix} = \begin{bmatrix} T & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots & \mathbf{x}^{(p)} \\ \mathbf{u}^{(1)} & \mathbf{u}^{(2)} & \dots & \mathbf{u}^{(p)} \end{bmatrix}\tag{9.13}$$

In Eq. (9.13), the matrix I is an identity matrix of appropriate size. Simplifying Eq. (9.13) we have:

$$\begin{bmatrix} T\mathbf{x}^{(2)} & T\mathbf{x}^{(3)} & \dots & T\mathbf{x}^{(p+1)} \\ \mathbf{y}^{(1)} & \mathbf{y}^{(2)} & \dots & \mathbf{y}^{(p)} \end{bmatrix} = \begin{bmatrix} TA & TB \\ C & D \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots & \mathbf{x}^{(p)} \\ \mathbf{u}^{(1)} & \mathbf{u}^{(2)} & \dots & \mathbf{u}^{(p)} \end{bmatrix}\tag{9.14}$$

Now let us represent the right side of Eq. (9.14) in the same transformed space of the left side:

$$\begin{bmatrix} T\mathbf{x}^{(2)} & T\mathbf{x}^{(3)} & \dots \\ \mathbf{y}^{(1)} & \mathbf{y}^{(2)} & \dots \end{bmatrix} = \begin{bmatrix} TA & TB \\ C & D \end{bmatrix} \begin{bmatrix} T^{-1} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} T & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots \\ \mathbf{u}^{(1)} & \mathbf{u}^{(2)} & \dots \end{bmatrix} \quad (9.15)$$

Rearranging the above equation provides us with the crucial observation that the same input-output data can be generated with very different parameters TAT^{-1} , TB , CT^{-1} , and D , and states $T\mathbf{x}$:

$$\begin{bmatrix} T\mathbf{x}^{(2)} & T\mathbf{x}^{(3)} & \dots & T\mathbf{x}^{(p+1)} \\ \mathbf{y}^{(1)} & \mathbf{y}^{(2)} & \dots & \mathbf{y}^{(p)} \end{bmatrix} = \begin{bmatrix} TAT^{-1} & TB \\ CT^{-1} & D \end{bmatrix} \begin{bmatrix} T\mathbf{x}^{(1)} & T\mathbf{x}^{(2)} & \dots & T\mathbf{x}^{(p)} \\ \mathbf{u}^{(1)} & \mathbf{u}^{(2)} & \dots & \mathbf{u}^{(p)} \end{bmatrix} \quad (9.16)$$

In comparing Eq. (9.16) with Eq. (9.12) we arrive at two important ideas: first, given an input-output sequence of data, it is not possible to estimate a unique set of parameters for Eq. (9.11) because there are an infinite number of equally valid candidates. Therefore, we need to change our objective and settle for finding one set of parameters among this infinite set. The second insight is that we can find a parameter set if we could estimate the state sequence \mathbf{x} in any arbitrary transformed space $T\mathbf{x}$. As we will show, when we pick this transformed space carefully, the problem lends itself to a closed form solution.

9.4 Subspace method: intuitive ideas

In Eq. (9.11), output \mathbf{y} is a linear function of the state \mathbf{x} and input \mathbf{u} . If we could somehow remove the effects of the sequence of inputs $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \dots, \mathbf{u}^{(p)}$ from the sequence of outputs $\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \dots, \mathbf{y}^{(p)}$, we would be left with a sequence that is a linear transformation on the sequence of states, i.e., $C\mathbf{x}^{(1)}, C\mathbf{x}^{(2)}, \dots, C\mathbf{x}^{(p)}$. If we had a linear transformation of the sequence of states, our problem would be trivial, as we could then recover the parameters that produced those states from Eq. (9.16). The subspace method is a geometric technique that precisely accomplishes this goal (van Overschee and De Moor, 1996).

When we project vector \mathbf{a} onto \mathbf{b} , we get a vector in the direction of \mathbf{b} with the magnitude of $\|\mathbf{a}\|\cos(\alpha)$, where $\|\mathbf{a}\|$ represents the length of vector \mathbf{a} and α is the angle between \mathbf{a} and \mathbf{b} . Using the expression \mathbf{a}/\mathbf{b} to represent this projection, we have:

$$\begin{aligned} \mathbf{a}/\mathbf{b} &\equiv \|\mathbf{a}\| \cos(\alpha) \frac{\mathbf{b}}{\|\mathbf{b}\|} \\ \cos(\alpha) &= \frac{\mathbf{a}^T \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|} \\ \mathbf{a}/\mathbf{b} &= \frac{\mathbf{a}^T \mathbf{b}}{\|\mathbf{b}\| \|\mathbf{b}\|} \mathbf{b} = \mathbf{a}^T \mathbf{b} (\mathbf{b}^T \mathbf{b})^{-1} \mathbf{b} \end{aligned} \quad (9.17)$$

Now suppose that we have an arbitrary matrix A of size 3×3 .

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \mathbf{a}_3^T \end{bmatrix} \quad (9.18)$$

Each row of this matrix is a vector in 3D space. If the row vectors are linearly independent (meaning that the rank of A is 3), then the row vectors can serve as a basis set to span 3D space. That is, we can construct any 3D vector as a linear combination of the row vectors of A . Now suppose we have another matrix B in which the row vectors are also in 3D space but the rank is 2.

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1^T \\ \mathbf{b}_2^T \end{bmatrix} \quad (9.19)$$

In this case, the space spanned by the row vectors of B is a plane. This plane defines a subspace of the space spanned by row vectors of A . When we project matrix A onto B , we are projecting the row vectors of A onto the subspace spanned by the row vectors of B . This is shown in Fig. 3A. To project vector \mathbf{a} onto matrix B , we have:

$$\mathbf{a}/B \equiv B^T (BB^T)^{-1} B \mathbf{a} \quad (9.20)$$

The result of Eq. (9.20) is a column vector. To represent it as a row vector we transpose it:

$$(\mathbf{a}/B)^T = \mathbf{a}^T B^T (BB^T)^{-1} B$$

To project matrix A onto matrix B we have:

$$A/B = \begin{bmatrix} (\mathbf{a}_1/B)^T \\ (\mathbf{a}_2/B)^T \\ (\mathbf{a}_3/B)^T \end{bmatrix} = AB^T (BB^T)^{-1} B \quad (9.21)$$

An important problem for us is to find the space that is perpendicular to the space defined by a matrix. The space spanned by the row vectors of B is simply a plane. Therefore, the space

perpendicular to this plane is simply a line (Fig. 3B). Let us use the term B^\perp to refer to the space perpendicular to the row vectors of B . Projecting A onto B^\perp we have:

$$A/B^\perp = A \left(I - B^T (BB^T)^{-1} B \right) \quad (9.22)$$

It then follows that $B/B^\perp = 0$.

Returning to our problem, in Eq. (9.11) we see that vector $\mathbf{y}^{(n)}$ is a linear combination of vectors $\mathbf{x}^{(n)}$ and $\mathbf{u}^{(n)}$, which means that vectors $\mathbf{x}^{(n)}$ and $\mathbf{u}^{(n)}$ are the bases for vector $\mathbf{y}^{(n)}$. By projecting the vector $\mathbf{y}^{(n)}$ onto a space perpendicular to $\mathbf{u}^{(n)}$, we will be left with a vector in the subspace spanned by $\mathbf{x}^{(n)}$. We will not recover $\mathbf{x}^{(n)}$ as a result of this projection, but we will recover a vector proportional to $\mathbf{x}^{(n)}$. As we noted in Eq. (9.16), our problem has no unique solution any ways, so we have no desire to recover $\mathbf{x}^{(n)}$. All we need to do is recover a vector proportional to it. If we do so, we can produce a generative model that precisely replicates the inputs and outputs of the system in question.

9.5 Subspace analysis

Our plan of attack is as follows. Suppose we construct matrices U and Y so that we have a compact way to represent the history of the inputs that we gave and outputs that we measured. We know that each row of matrix Y lives in the space spanned by the row vectors in X (history of hidden states, which is unknown to us), and U (history of inputs that we gave). We will project Y onto the subspace perpendicular to U , written as U^\perp . By doing so, we will get rid of the contribution of U , leaving only the components of Y that live in the subspace described by X . Therefore, by projecting Y onto U^\perp , we will end up with a new matrix that is precisely equal to CX . Because C is a constant matrix, we will in fact recover the history of the hidden states up to a constant ‘multiple’.

The first step is to arrange the input and output data in what is called a *Hankel* matrix as follows:

$$\begin{aligned}
Y_{1|i} &\equiv \begin{bmatrix} \mathbf{y}^{(1)} & \mathbf{y}^{(2)} & \dots & \mathbf{y}^{(j)} \\ \mathbf{y}^{(2)} & \mathbf{y}^{(3)} & \dots & \mathbf{y}^{(j+1)} \\ \vdots & \vdots & & \vdots \\ \mathbf{y}^{(i)} & \mathbf{y}^{(i+1)} & \dots & \mathbf{y}^{(i+j-1)} \end{bmatrix} \\
U_{1|i} &\equiv \begin{bmatrix} \mathbf{u}^{(1)} & \mathbf{u}^{(2)} & \dots & \mathbf{u}^{(j)} \\ \mathbf{u}^{(2)} & \mathbf{u}^{(3)} & \dots & \mathbf{u}^{(j+1)} \\ \vdots & \vdots & & \vdots \\ \mathbf{u}^{(i)} & \mathbf{u}^{(i+1)} & \dots & \mathbf{u}^{(i+j-1)} \end{bmatrix}
\end{aligned} \tag{9.23}$$

In these matrices, $i \ll j$, i.e., number of rows is much smaller than number of columns. If we now label the (unknown) state sequence as matrix X ,

$$X_i \equiv \begin{bmatrix} \mathbf{x}^{(i)} & \mathbf{x}^{(i+1)} & \dots & \mathbf{x}^{(i+j-1)} \end{bmatrix} \tag{9.24}$$

we see that by projecting the row vectors of Y onto the subspace spanned by U^\perp , we can recover matrix CX . That is, we can recover the subspace spanned by the state vectors. To show this, let us write the output matrix Y as a linear function of states and inputs:

$$Y_{1|i} = \begin{bmatrix} C\mathbf{x}^{(1)} + D\mathbf{u}^{(1)} & C\mathbf{x}^{(2)} + D\mathbf{u}^{(2)} & \dots \\ CA\mathbf{x}^{(1)} + CB\mathbf{u}^{(1)} + D\mathbf{u}^{(2)} & CA\mathbf{x}^{(2)} + CB\mathbf{u}^{(2)} + D\mathbf{u}^{(3)} & \dots \\ CA^2\mathbf{x}^{(1)} + CAB\mathbf{u}^{(1)} + CB\mathbf{u}^{(2)} + D\mathbf{u}^{(3)} & CA^2\mathbf{x}^{(2)} + CAB\mathbf{u}^{(2)} + CB\mathbf{u}^{(3)} + D\mathbf{u}^{(4)} & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

We will now attempt to write the history of measurements $Y_{1|i}$ as a linear function of history of states X_i and history of input $U_{1|i}$. If we define matrices Γ_i and H_i as follows:

$$\begin{aligned}
\Gamma_i &\equiv \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{i-1} \end{bmatrix} \\
H_i &\equiv \begin{bmatrix} D & 0 & 0 & \dots & 0 \\ CB & D & 0 & \dots & 0 \\ CAB & CB & D & & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ CA^{i-2}B & CA^{i-3}B & \dots & CB & D \end{bmatrix}
\end{aligned} \tag{9.25}$$

then we can write:

$$Y_{1|i} = \Gamma_i X_1 + H_i U_{1|i} \tag{9.26}$$

Eq. (9.26) represents a short-hand way of writing the relationship between the history of inputs, the history of hidden states, and the history of observations. It is useful to use a similar notation to write the relationship between the history of states and the history of inputs. To do so, we note the following:

$$\begin{aligned}
\mathbf{x}^{(2)} &= A\mathbf{x}^{(1)} + B\mathbf{u}^{(1)} \\
\mathbf{x}^{(3)} &= A^2\mathbf{x}^{(1)} + AB\mathbf{u}^{(1)} + B\mathbf{u}^{(2)} \\
\mathbf{x}^{(4)} &= A^3\mathbf{x}^{(1)} + A^2B\mathbf{u}^{(1)} + AB\mathbf{u}^{(2)} + B\mathbf{u}^{(3)} \\
\mathbf{x}^{(i+1)} &= A^i\mathbf{x}^{(1)} + A^{i-1}B\mathbf{u}^{(1)} + A^{i-2}B\mathbf{u}^{(2)} + \dots + B\mathbf{u}^{(i)}
\end{aligned} \tag{9.27}$$

If we now define matrix Δ_i as follows:

$$\Delta_i \equiv \begin{bmatrix} A^{i-1}B & A^{i-2}B & \dots & B \end{bmatrix} \tag{9.28}$$

we can write the state update equation as:

$$X_{i+1} = A^i X_1 + \Delta_i U_{1|i} \tag{9.29}$$

We begin our campaign with Eq. (9.26). Re-arranging this equation we have:

$$X_1 = \Gamma_i^* Y_{1|i} - \Gamma_i^* H_i U_{1|i} \tag{9.30}$$

The superscript * in Eq. (9.30) indicates a pseudo-inverse. Inserting the above representation into Eq. (9.29) we have:

$$X_{i+1} = A^i \Gamma_i^* Y_{1|i} - A^i \Gamma_i^* H_i U_{1|i} + \Delta_i U_{1|i} \tag{9.31}$$

In the above equation, we know $Y_{1|i}$ and $U_{1|i}$ but nothing else. Let us re-write the above equation in terms of things that we know and things that we do not know. If we label things that we know with matrix $W_{1|i}$ as follows:

$$W_{1|i} \equiv \begin{bmatrix} U_{1|i} \\ Y_{1|i} \end{bmatrix} \tag{9.32}$$

and then define matrix L_i as follows:

$$L_i \equiv \begin{bmatrix} \Delta_i - A^i \Gamma_i^* H_i & A^i \Gamma_i^* \end{bmatrix} \tag{9.33}$$

we can now write the history of the states as a linear function of things that we know:

$$X_{i+1} = L_i W_{1|i} \tag{9.34}$$

From Eq. (9.26) we have:

$$Y_{i+1|2i} = \Gamma_i X_{i+1} + H_i U_{i+1|2i} \tag{9.35}$$

Inserting Eq. (9.34) into above we have:

$$Y_{i+1|2i} = \Gamma_i L_i W_{1|i} + H_i U_{i+1|2i} \quad (9.36)$$

Now we project our history of observations $Y_{i+1|2i}$ onto the subspace perpendicular to our history of inputs $U_{i+1|2i}^\perp$:

$$Y_{i+1|2i} / U_{i+1|2i}^\perp = \Gamma_i L_i W_{1|i} / U_{i+1|2i}^\perp + H_i U_{i+1|2i} / U_{i+1|2i}^\perp \quad (9.37)$$

As the second term on the right side of Eq. (9.37) is zero, we have:

$$Y_{i+1|2i} / U_{i+1|2i}^\perp = \Gamma_i L_i W_{1|i} / U_{i+1|2i}^\perp \quad (9.38)$$

Note that in the above equations, we know everything except $\Gamma_i L_i$. Let us re-arrange this equation and put our known quantities on the right side and the unknowns on the left side:

$$\Gamma_i L_i = \left[Y_{i+1|2i} / U_{i+1|2i}^\perp \right] \left[W_{1|i} / U_{i+1|2i}^\perp \right]^* \quad (9.39)$$

Now if we simply multiply both side with another known quantity $W_{1|i}$, we have:

$$\Gamma_i L_i W_{1|i} = \left[Y_{i+1|2i} / U_{i+1|2i}^\perp \right] \left[W_{1|i} / U_{i+1|2i}^\perp \right]^* W_{1|i} \quad (9.40)$$

The left side of the above equation can be simplified via Eq. (9.34):

$$\Gamma_i X_{i+1} = \left[Y_{i+1|2i} / U_{i+1|2i}^\perp \right] \left[W_{1|i} / U_{i+1|2i}^\perp \right]^* W_{1|i} \quad (9.41)$$

The right side of Eq. (9.41) includes quantities that are all known to us, and so we can compute them. Let us label this matrix as O_{i+1} :

$$O_{i+1} \equiv \left[Y_{i+1|2i} / U_{i+1|2i}^\perp \right] \left[W_{1|i} / U_{i+1|2i}^\perp \right]^* W_{1|i} \quad (9.42)$$

The term on the left side of Eq. (9.41) is simply a linear transformation of the states that we are looking for:

$$O_{i+1} = \Gamma_i X_{i+1} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{i-1} \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(i+1)} & \mathbf{x}^{(i+2)} & \dots & \mathbf{x}^{(i+j)} \end{bmatrix} \quad (9.43)$$

At this point we can compute the matrix O_{i+1} . Our final step is to factor it so that we recover a matrix \hat{X}_{i+1} . Our estimate \hat{X}_{i+1} will not be equal to X_{i+1} , but it will be related to it by a linear transformation. If we do a singular value decomposition of matrix O_{i+1} , we have:

$$\begin{aligned}
O_{i+1} &= PSV \\
&= \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_i \end{bmatrix} \begin{bmatrix} s_1 & 0 & \cdots & 0 \\ 0 & s_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & s_n \end{bmatrix} \begin{bmatrix} \mathbf{v}^{(1)} & \mathbf{v}^{(2)} & \cdots & \mathbf{v}^{(j)} \end{bmatrix}
\end{aligned} \tag{9.44}$$

In this decomposition, the dimensions of matrix P_i will be the same as matrix C , which is the same as the dimensions of matrices CA , CA^2 , etc. The dimensions of the matrix S is $n \times n$, where n is the dimension of the state vector \mathbf{x} . The crucial idea is that the number of singular values associated with matrix O_{i+1} is the size of the state vector that we are seeking. While we cannot recover the state matrix X_{i+1} , we can recover something that is a linear transformation of it. To see this, say that matrix T is an arbitrary and unknown invertible matrix of appropriate size. The singular value decomposition of O_{i+1} can be written as:

$$\Gamma_i X_{i+1} = PSV = PS^{1/2} T T^{-1} S^{1/2} V \tag{9.45}$$

In Eq. (9.45), the term $S^{1/2}$ is the ‘square-root’ of the matrix S , which in this case is simply a matrix with the square root of each diagonal term. The state matrix X_{i+1} that we are looking for is related to the singular value decomposition matrices as follows:

$$X_{i+1} = T^{-1} S^{1/2} V \tag{9.46}$$

We do not know matrix T . So if we simply set our state estimate as:

$$\hat{X}_{i+1} = S^{1/2} V \tag{9.47}$$

then our state estimate will be a linear transformation on the actual states:

$$\hat{X}_{i+1} = T X_{i+1} \tag{9.48}$$

Having found an estimate of the hidden states \hat{X}_{i+1} , the problem of finding the parameters of the system (matrices A , B , C , D) is now trivial. If we define the matrix V without its last column as:

$$V| \equiv \begin{bmatrix} \mathbf{v}^{(1)} & \mathbf{v}^{(2)} & \cdots & \mathbf{v}^{(j-1)} \end{bmatrix} \tag{9.49}$$

and the matrix V without its first column as:

$$|V \equiv \begin{bmatrix} \mathbf{v}^{(2)} & \mathbf{v}^{(3)} & \cdots & \mathbf{v}^{(j)} \end{bmatrix} \tag{9.50}$$

and similarly for state estimates as:

$$\begin{aligned}
\hat{X}_{i+2}| &= S^{1/2} |V \\
\hat{X}_{i+1}| &= S^{1/2} V|
\end{aligned} \tag{9.51}$$

and similarly for input and output matrices as:

$$\begin{aligned} Y_{i+1} &\equiv \begin{bmatrix} \mathbf{y}^{(i+1)} & \mathbf{y}^{(i+2)} & \dots & \mathbf{y}^{(j+i-1)} \end{bmatrix} \\ U_{i+1} &\equiv \begin{bmatrix} \mathbf{u}^{(i+1)} & \mathbf{u}^{(i+2)} & \dots & \mathbf{u}^{(j+i-1)} \end{bmatrix} \end{aligned} \quad (9.52)$$

we will have a simple linear equation:

$$\begin{bmatrix} \hat{X}_{i+2} \\ Y_{i+1} \end{bmatrix} = \begin{bmatrix} \hat{A} & \hat{B} \\ \hat{C} & \hat{D} \end{bmatrix} \begin{bmatrix} \hat{X}_{i+1} \\ U_{i+1} \end{bmatrix} \quad (9.53)$$

to solve for the unknown parameters A, B, C, D :

$$\begin{bmatrix} \hat{A} & \hat{B} \\ \hat{C} & \hat{D} \end{bmatrix} = \begin{bmatrix} \hat{X}_{i+2} \\ Y_{i+1} \end{bmatrix} \begin{bmatrix} \hat{X}_{i+1} \\ U_{i+1} \end{bmatrix}^* \quad (9.54)$$

We can use the following to find the initial state:

$$\hat{\mathbf{x}}^{(1)} = \hat{A}^{-i} \hat{\mathbf{x}}^{(i+1)} - \hat{A}^{-1} \hat{B} \mathbf{u}^{(1)} - \hat{A}^{-2} \hat{B} \mathbf{u}^{(2)} - \dots - \hat{A}^{-i} \hat{B} \mathbf{u}^{(i)} \quad (9.55)$$

Now the important point to note is that while our system with parameters \hat{A} , \hat{B} , \hat{C} , and \hat{D} will be indistinguishable from the original system with parameters A, B, C, D (in terms of input-output behavior), our estimates are not the same value as the system parameters. They are related by an unknown linear transformation as defined by matrix T :

$$\begin{bmatrix} \hat{A} & \hat{B} \\ \hat{C} & \hat{D} \end{bmatrix} = \begin{bmatrix} TAT^{-1} & TB \\ CT^{-1} & D \end{bmatrix} \quad (9.56)$$

In summary, the problem of structural learning is that of describing a dynamical system that in principle can accurately predict the sensory consequences of motor commands, i.e., learn the structure of a forward model. Even for a linear system without noise, this problem has no unique solution. However, we can find one particular solution by projecting the history of our measurements upon a subspace that is perpendicular to the space defined by the history of our motor commands. Therefore, a fundamental part of structuring learning is to keep a history of the motor commands so that one can build this subspace. Once the measurements (sensory observations) are projected onto this subspace, the result is a linear transformation of the hidden states of the system that we wish to model.

9.6 Examples

Consider a system with the following dynamics:

$$\begin{aligned}
x^{(n+1)} &= 0.75x^{(n)} + 0.3u^{(n)} \\
y^{(n)} &= 0.5x^{(n)} \\
x^{(1)} &= 0.0
\end{aligned} \tag{9.57}$$

Suppose that we give a sequence of square-wave inputs $\{u^{(1)}, u^{(2)}, \dots, u^{(p)}\}$ to this system, as shown in the top row of Fig. 9.4A, and observe the sequence of outputs $\{y^{(1)}, y^{(2)}, \dots, y^{(p)}\}$, as shown in the bottom row of Fig. 9.4A. From this input/output data, we form the $Y_{\lfloor i\rfloor}$ and $U_{\lfloor i\rfloor}$ matrices in Eq. (9.23) (in our example here, we set $i = 4$), and then the matrix O_{i+1} from Eq. (9.42). The singular value decomposition of this matrix provides a single singular value, which specifies that the size of the state vector is 1, and the following estimate of the structure for this system:

$$\begin{aligned}
\hat{A} &= 0.75 & \hat{B} &= -0.09 & \hat{x}^{(1)} &= 0 \\
\hat{C} &= -1.7 & \hat{D} &= 0
\end{aligned} \tag{9.58}$$

Notice that we did not recover the system parameters. Despite this, the system with the parameters in Eq. (9.58) is identical to the one in Eq. (9.57). For example, if we give the input $\{u^{(1)}, u^{(2)}, \dots, u^{(p)}\}$ to our system with parameters in Eq. (9.58), we find $\sum_n (\hat{y}^{(n)} - y^{(n)})^2 = 0$.

The system in Eq. (9.57) was relatively simple in that there was only a single state and our measurement on each trial was proportional to that state. To make our problem more interesting, suppose that we have a system with many states, and that we can only observe the sum of these states:

$$\begin{aligned}
\mathbf{x}^{(n+1)} &= \begin{bmatrix} 0.9 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.2 \end{bmatrix} \mathbf{x}^{(n)} + \begin{bmatrix} 0.3 \\ 0.5 \\ 0.8 \end{bmatrix} u^{(n)} \\
y^{(n)} &= [1 \quad 1 \quad 1] \mathbf{x}^{(n)} \\
\mathbf{x}^{(1)} &= [0.1 \quad -0.1 \quad 0]^T
\end{aligned} \tag{9.59}$$

The inputs to this system and the outputs are shown in Fig. (9.4B). Superficially, the response in Fig. (9.4B) is similar to that in Fig. (9.4A), except perhaps for a slower time-constant. However, when we run the algorithm (we set $i = 5$ in Eq. 9.23), we see that the matrix O_{i+1} has 3 singular values, and we arrive at the following estimate for the structure of the system:

$$\begin{bmatrix} \hat{A} & \hat{B} \\ \hat{C} & \hat{D} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 0.659 & -0.303 & 0.014 \\ -0.260 & 0.549 & 0.134 \\ 0.070 & 0.191 & 0.393 \\ -4.53 & -1.47 & 0.073 \end{bmatrix} & \begin{bmatrix} -0.302 \\ -0.155 \\ -0.063 \\ 0 \end{bmatrix} \end{bmatrix} \quad \hat{\mathbf{x}}^{(1)} = \begin{bmatrix} -0.011 \\ 0.0354 \\ 0.0375 \end{bmatrix} \quad (9.60)$$

In Eq. (9.60), \hat{A} is a 3x3 matrix, \hat{B} is 3x1, \hat{C} is 1x3, and \hat{D} is a scalar. Our estimate produces an exact match to the measured data: $\sum_n (\hat{y}^{(n)} - y^{(n)})^2 = 0$.

Finally, let us consider a system that is driven by random inputs $u^{(n)}$ rather than a square wave. An example is shown in Fig. (9.4C). The dynamics of this system are as follows:

$$\begin{aligned} \mathbf{x}^{(k+1)} &= \begin{bmatrix} 0.9 & 0 \\ 0 & 0.5 \end{bmatrix} \mathbf{x}^{(k)} + \begin{bmatrix} 0.1 \\ 0.3 \end{bmatrix} u^{(k)} \\ y^{(k)} &= [1 \quad 1] \mathbf{x}^{(k)} \\ \mathbf{x}^{(1)} &= [0.1 \quad -0.1]^T \end{aligned} \quad (9.61)$$

Subspace analysis uncovers two singular values, and provides the following estimate for the structure of this system:

$$\begin{bmatrix} \hat{A} & \hat{B} \\ \hat{C} & \hat{D} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 0.657 & -0.241 & -0.267 \\ -0.158 & 0.743 & -0.029 \\ 1.46 & 0.319 & 0 \end{bmatrix} & \begin{bmatrix} -0.267 \\ -0.029 \\ 0 \end{bmatrix} \end{bmatrix} \quad \hat{\mathbf{x}}^{(1)} = \begin{bmatrix} 0.028 \\ -0.127 \end{bmatrix} \quad (9.62)$$

and the resulting output of the estimated system exactly matches the measured data.

The systems that we arrived at in Eq. (9.60) and (9.62) may produce the same outputs for a sequence of inputs as that in Eq. (9.59) and (9.61), but they are much harder to interpret. For example, the system in Eq. (9.59) has three states, each that decays with a different rate, with no interaction between these states (i.e., the matrix A is diagonal). Furthermore, the observation y is simply the sum of these states, in which the states are weighted equally. In contrast, our estimate in Eq. (9.60) is a system with rather complicated interactions between the states, with observation y that weights the states unequally. It seems hard to believe that these two systems are really the same. To show that they are, let us find the matrix T in Eq. (9.56) that transforms one set of parameters into the other. The key point is the relationship between A and \hat{A} :

$$\hat{A} = TAT^{-1} \quad (9.63)$$

In our original system in Eq. (9.59), the matrix A was diagonal. How do we factor \hat{A} into the form shown in Eq. (9.61) such that the matrix A is diagonal? We proceed by finding the eigen vectors and eigen values of matrix \hat{A} . Suppose that \hat{A} has eigen values $\lambda_1, \lambda_2, \dots, \lambda_k$, and eigen vectors:

$$\mathbf{q}_1 = \begin{bmatrix} q_{11} \\ \vdots \\ q_{1k} \end{bmatrix}, \mathbf{q}_2 = \begin{bmatrix} q_{21} \\ \vdots \\ q_{2k} \end{bmatrix}, \dots \quad (9.64)$$

Let us arrange the eigen vectors and values in matrix form:

$$Q \equiv [\mathbf{q}_1 \quad \mathbf{q}_2 \quad \dots \quad \mathbf{q}_k] \\ L \equiv \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_k \end{bmatrix} \quad (9.65)$$

Multiplying the matrix \hat{A} by its eigen vectors gives us:

$$\begin{aligned} \hat{A}Q &= [\hat{A}\mathbf{q}_1 \quad \hat{A}\mathbf{q}_2 \quad \dots \quad \hat{A}\mathbf{q}_k] \\ &= [\lambda_1\mathbf{q}_1 \quad \lambda_2\mathbf{q}_2 \quad \dots \quad \lambda_k\mathbf{q}_k] \end{aligned} \quad (9.66)$$

We can factor the matrix $\hat{A}Q$ as follows:

$$\begin{aligned} \hat{A}Q &= \begin{bmatrix} \lambda_1 q_{11} & \lambda_2 q_{21} & \dots & \lambda_k q_{k1} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_1 q_{1k} & \lambda_2 q_{2k} & \dots & \lambda_k q_{kk} \end{bmatrix} \\ &= \begin{bmatrix} q_{11} & q_{21} & \dots & q_{k1} \\ \vdots & \vdots & \ddots & \vdots \\ q_{1k} & q_{2k} & \dots & q_{kk} \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_k \end{bmatrix} \end{aligned} \quad (9.67)$$

which we can summarize as:

$$\hat{A}Q = QL \quad (9.68)$$

We finally arrive at a factored form of \hat{A} in terms of a diagonal matrix L :

$$\hat{A} = QLQ^{-1} \quad (9.69)$$

So if we set $T = Q$, where Q is the matrix containing the eigen vectors of \hat{A} , we can transform our estimated parameter \hat{A} to a form that is diagonal, producing another equivalent system to the one that originally produced the input/output data. Note that once again we cannot find the

parameters of the original system, but we can find parameters that make a system that is indistinguishable from the original one from the point of view of dynamics.

9.7 Estimating the noise

In the case that our system has noise, in the form:

$$\begin{aligned} \mathbf{x}^{(n+1)} &= A\mathbf{x}^{(n)} + B\mathbf{u}^{(n)} + \boldsymbol{\varepsilon}_x^{(n)} & \boldsymbol{\varepsilon}_x &\square N(0, Q) \\ \mathbf{y}^{(n)} &= C\mathbf{x}^{(n)} + D\mathbf{u}^{(n)} + \boldsymbol{\varepsilon}_y^{(n)} & \boldsymbol{\varepsilon}_y &\square N(0, R) \end{aligned} \quad (9.70)$$

the key step of determining the size of the hidden states by examining the number of singular values in O_{i+1} will need to be slightly modified. In principle, the number of singular values will be equal to i in Eq. (9.23). That is, once the system has noise, we can no longer identify with absolute certainty the size of the hidden state vector. In practice, the singular values need to be examined and hopefully most will be rather small and can be disregarded. Once the parameters \hat{A} , \hat{B} , \hat{C} , and \hat{D} are estimated, the noise variance Q and R can be computed from the residuals in the fit. If we define the residual in the estimate of state as in Eq. (9.71), then state noise is the variance of this estimate:

$$\begin{aligned} \tilde{\mathbf{x}}^{(n)} &= \hat{\mathbf{x}}^{(n)} - \hat{A}\hat{\mathbf{x}}^{(n-1)} - \hat{B}\mathbf{u}^{(n-1)} \\ Q &= E\left[(\tilde{\mathbf{x}} - E[\tilde{\mathbf{x}}])(\tilde{\mathbf{x}} - E[\tilde{\mathbf{x}}])^T \right] \end{aligned} \quad (9.71)$$

Similarly, measurement noise is the variance of the residual in the estimate of output:

$$\begin{aligned} \tilde{\mathbf{y}}^{(n)} &= \mathbf{y}^{(n)} - \hat{\mathbf{y}}^{(n)} \\ R &= E\left[(\tilde{\mathbf{y}} - E[\tilde{\mathbf{y}}])(\tilde{\mathbf{y}} - E[\tilde{\mathbf{y}}])^T \right] \end{aligned} \quad (9.72)$$

9.8 Identifying the structure of the learner

There is a practical application for the framework that we just developed: we can use it to model the learner. For example, say that we have collected some data from a subject during some behavioral task in which she made movements and adapted to a perturbation. We imagine that our subject has a state on each trial and based on this state programs a motor command (which we can record based on their movements). As a consequence of their motor command, our subject makes an observation (e.g., the limb did not move in the predicted direction), and then learns from the resulting prediction error. Our data set consists of a series of trials in which we have

given perturbations, and a series of movements (or predictions) that the subject has made. We can represent the learner as a dynamical system in which her states change as a consequence of her prediction errors. We are interested in quantifying the timescales of these states, and sensitivity of each state to a prediction error.

Let us apply this idea to a simple task: saccade adaptation. A subject is provided with a target, makes a saccade to that target, and during that saccade the target is moved to a new location. As the saccade terminates, the subject observes that the target is not on the fovea, that is, there is a prediction error. Trial after trial, the subject learns to alter the magnitude of the saccade to minimize this error. Suppose that the state of the learner can be represented with vector \mathbf{x} (of unknown dimensionality). The state is affected by three factors: visual error \tilde{y} at end of a saccade, passage of time between trials, and Gaussian noise $\boldsymbol{\varepsilon}_x$. If we assume that the inter-trial interval is constant, then we can write the change in states of the learner as:

$$\mathbf{x}^{(n+1)} = A\mathbf{x}^{(n)} + \mathbf{b}\tilde{y}^{(n)} + \boldsymbol{\varepsilon}_x \quad (9.73)$$

In this equation, the matrix A specifies how the states will change from trial n to $n+1$ because of passage of time, and the vector \mathbf{b} specifies how the states will change because of the error observed on trial n . We cannot directly observe the states, but can measure saccade amplitude on trial n as $y^{(n)}$. Let us assume that saccade amplitude $y^{(n)}$ is affected by target location $p^{(n)}$, some inherent bias that the subject may have y_b , the state of the subject $\mathbf{x}^{(n)}$, plus noise $\boldsymbol{\varepsilon}_y$ inherent in the execution of the movement. This is written as:

$$y^{(n)} = p^{(n)} - y_b + \mathbf{c}^T \mathbf{x}^{(n)} + \boldsymbol{\varepsilon}_y \quad (9.74)$$

In Eq. (9.74), the vector \mathbf{c} specifies the relative weight of each state in influencing the saccade amplitude.

Like any system identification problem, we want to give ‘inputs’ to our learner and then measure her behavior. These inputs are in the form of a perturbation, i.e., we will move the target $p^{(n)}$ by amount $u^{(n)}$ during the saccade, so that when the eye movement completes, there will be some endpoint errors. The error on that trial will be:

$$\begin{aligned} \tilde{y}^{(n)} &= p^{(n)} + u^{(n)} - y^{(n)} - y_b \\ &= u^{(n)} - \mathbf{c}^T \mathbf{x}^{(n)} - \boldsymbol{\varepsilon}_y \end{aligned} \quad (9.75)$$

Inserting Eq. (9.75) into Eq. (9.73) produces our state space model of the saccade adaptation task:

$$\begin{aligned}\mathbf{x}^{(n+1)} &= (\mathbf{A} - \mathbf{b}\mathbf{c}^T)\mathbf{x}^{(n)} + \mathbf{b}(u^{(n)} - \varepsilon_y) + \varepsilon_x & \varepsilon_x &\square N(0, Q) \\ y^{(n)} &= p^{(n)} - y_b + \mathbf{c}^T\mathbf{x}^{(n)} + \varepsilon_y & \varepsilon_y &\square N(0, r)\end{aligned}\quad (9.76)$$

In a typical experiment, for each subject we give a sequence of targets $p^{(n)}$ and on each trial displace that target during the saccade by amount $u^{(n)}$. We then measure the saccade amplitude $y^{(n)}$. Our objective is to find the structure of the learner, i.e., parameters \mathbf{A} , \mathbf{b} , and \mathbf{c} in Eq. (9.76).

Vincent Ethier, David Zee, and Shadmehr (Ethier et al., 2008) performed a saccade adaptation experiment and then used subspace analysis to estimate the structure of the learner from trial-to-trial movement data. In the experiment, they considered two kinds of trials: perturbation trials in which the target of the saccade was moved during the saccade, and *error-clamp* trials in which errors were eliminated by moving the target so that it was located on the fovea at the endpoint of the saccade. In adaptation trials, $u^{(n)}$ was the intra-saccadic target displacement. In error-clamp trials

$$u^{(n)} = y^{(n)} - p^{(n)} + y_b \quad (9.77)$$

The paradigm is summarized in Fig. 9.5A. The experiment began with error-clamp trials, was followed by a gain-down session, followed by a gain-up session ('extinction'), and finally error-clamp trials. The objective was to unmask the multiple timescales of memory and test for 'spontaneous recovery'. In summary, the mathematical problem consisted in finding the parameters of the dynamical system in Eq. (9.76), given a sequence of inputs $u^{(n)}$ (target displacements) and measurements $y^{(n)}$ (saccade amplitudes).

The averaged saccade amplitudes produced by a group of subjects is shown in Fig. 9.5B. This averaged data was analyzed using subspace methods and parameters for the resulting system were identified. Using these parameters, a state estimation procedure (Kalman filter) was then used to estimate the two hidden states of the system. The model's performance is shown in Fig. 9.5C and the estimated states of the system (the fast and slow states) are plotted in Fig. 9.5D. The fast state shows rapid adaptation at set starts, and forgetting at set breaks. Of interest is the fact that the slow state shows no forgetting at set breaks, and little or no unlearning during extinction.

A particularly useful way to visualize the parameters of the model is one in which the states are assumed to be independent, i.e., the transition matrix $A - \mathbf{b}\mathbf{c}^T$ is diagonal. This would produce a time constant of forgetting for each state. Suppose that we represent the state equation in continuous time:

$$\dot{\mathbf{x}}(t) = A_c \mathbf{x}(t) + \mathbf{b}_c u(t) + \pi_x \quad (9.78)$$

where $A_c = \Delta^{-1}(A - \mathbf{b}\mathbf{c}^T - I)$, $\mathbf{b}_c = \mathbf{b}\Delta^{-1}$, and Δ is the inter-trial interval, set to 1250ms. If we represent vector \mathbf{x} as $[x_f, x_s]^T$, i.e., the fast and slow states, then λ_s and λ_f refer to the time constant of the solution to this differential equation. The fit to the group data produced a decay time constant of $\lambda_f = 28$ sec for the fast state and $\lambda_s = 7$ min for the slow state. Error sensitivity of the fast state was 18 times larger for the fast state, i.e., $b_f/b_s = 18$. Finally, saccade amplitudes relied almost twice as much on the slow than the fast state, i.e., $c_s/c_f = 1.7$.

9.9 Expectation Maximization (EM)

Among people who study control theory, the subspace approach is well known and often used for system identification. In the machine learning community, however, this approach is less used. Instead, an algorithm called Expectation Maximization (EM) (Shumway and Stoffer, 1982) is often employed for solving the problem of system identification (Ghahramani and Hinton, 1996). Here, we briefly introduce EM as it has been used effectively to represent the human learner in terms of a linear state-space model (Cheng and Sabes, 2006).

Given some sequence of inputs $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \dots$ and a sequence of measurements $\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \dots$, our objective is to find the structure (matrices A, B, C , etc.) of the linear system:

$$\begin{aligned} \mathbf{x}^{(n+1)} &= A\mathbf{x}^{(n)} + B\mathbf{u}^{(n)} + \boldsymbol{\varepsilon}_x^{(n)} & \boldsymbol{\varepsilon}_x &\square N(0, Q) \\ \mathbf{y}^{(n)} &= C\mathbf{x}^{(n)} + \boldsymbol{\varepsilon}_y^{(n)} & \boldsymbol{\varepsilon}_y &\square N(0, R) \end{aligned} \quad (9.79)$$

Unlike subspace analysis, here we will assume that we know the dimensionality of the hidden states \mathbf{x} . There are two kinds of unknowns in our model: the structural quantities

$\theta = \{A, B, C, Q, R, \hat{\mathbf{x}}^{(0)}, P^{(0)}\}$ (where the last two terms are the prior estimate of state and its

variance), and the hidden states $\hat{\mathbf{x}}^{(1)}, \hat{\mathbf{x}}^{(2)}, \dots$. If we knew the structure θ , then finding the hidden states would be easy (via the Kalman filter, for example). If we knew the hidden states, then

finding the structural quantities would be easy (via maximizing a likelihood measure of the observe data). EM proceeds by performing these two steps in sequence, and repeats until the parameters converge.

The starting point in EM is to describe the expected complete log-likelihood. In our problem, this is the joint probability of the hidden states $\{\mathbf{x}\}_1^N = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)}\}$ and the observed quantities $\{\mathbf{y}\}_1^N = \{\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \dots, \mathbf{y}^{(N)}\}$, given the inputs $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \dots$ and structural parameters θ . The expected complete log-likelihood that we are after is defined as:

$$\square \equiv E \left[\log p(\{\mathbf{x}\}, \{\mathbf{y}\} | \theta, \{\mathbf{u}\}) \right] \quad (9.80)$$

In the E step, we fix θ and try to maximize the expected complete log-likelihood by setting expected value of our states $\{\mathbf{x}\}_1^N$ to their posterior probabilities (done via a Kalman filter). In the M step, we fix the expected value of our states $\{\mathbf{x}\}_1^N$ and try to maximize the expected complete log-likelihood by estimating the parameters θ .

To compute the expected complete log-likelihood, let us start with the following equality:

$$p(\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \mathbf{y}^{(1)} | \mathbf{u}^{(0)}) = p(\mathbf{y}^{(1)} | \mathbf{x}^{(1)}, \mathbf{x}^{(0)}, \mathbf{u}^{(0)}) p(\mathbf{x}^{(1)}, \mathbf{x}^{(0)} | \mathbf{u}^{(0)}) \quad (9.81)$$

Using Eq. (9.79), the above equation can be simplified to:

$$p(\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \mathbf{y}^{(1)} | \mathbf{u}^{(0)}) = p(\mathbf{y}^{(1)} | \mathbf{x}^{(1)}) p(\mathbf{x}^{(1)} | \mathbf{x}^{(0)}, \mathbf{u}^{(0)}) p(\mathbf{x}^{(0)}) \quad (9.82)$$

Similarly, we have:

$$\begin{aligned} p(\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{y}^{(1)}, \mathbf{y}^{(2)} | \mathbf{u}^{(1)}, \mathbf{u}^{(0)}) &= p(\mathbf{y}^{(1)}, \mathbf{y}^{(2)} | \{\mathbf{x}\}_0^2, \{\mathbf{u}\}_0^1) p(\{\mathbf{x}\}_0^2 | \{\mathbf{u}\}_0^1) \\ &= p(\mathbf{y}^{(2)} | \mathbf{y}^{(1)}, \{\mathbf{x}\}_0^2, \{\mathbf{u}\}_0^1) p(\mathbf{y}^{(1)} | \{\mathbf{x}\}_0^2, \{\mathbf{u}\}_0^1) \\ &\quad \times p(\mathbf{x}^{(2)} | \mathbf{x}^{(1)}, \mathbf{x}^{(0)}, \{\mathbf{u}\}_0^1) p(\mathbf{x}^{(1)}, \mathbf{x}^{(0)} | \{\mathbf{u}\}_0^1) \quad (9.83) \\ &= p(\mathbf{y}^{(2)} | \mathbf{x}^{(2)}) p(\mathbf{y}^{(1)} | \mathbf{x}^{(1)}) \\ &\quad \times p(\mathbf{x}^{(2)} | \mathbf{x}^{(1)}, \mathbf{u}^{(1)}) p(\mathbf{x}^{(1)} | \mathbf{x}^{(0)}, \mathbf{u}^{(0)}) p(\mathbf{x}^{(0)}) \end{aligned}$$

So we can conclude that:

$$p(\{\mathbf{x}\}_0^N, \{\mathbf{y}\}_1^N | \theta, \{\mathbf{u}\}_1^N) = \left[\prod_{n=1}^N p(\mathbf{y}^{(n)} | \mathbf{x}^{(n)}) \right] \left[\prod_{n=1}^N p(\mathbf{x}^{(n)} | \mathbf{x}^{(n-1)}, \mathbf{u}^{(n-1)}) \right] p(\mathbf{x}^{(0)}) \quad (9.84)$$

In Eq. (9.84), we have the following normal distributions:

$$\begin{aligned} p(\mathbf{y}^{(n)} | \mathbf{x}^{(n)}) &= N(\mathbf{C}\mathbf{x}^{(n)}, R) \\ p(\mathbf{x}^{(n+1)} | \mathbf{x}^{(n)}, \mathbf{u}^{(n)}) &= N(\mathbf{A}\mathbf{x}^{(n)} + \mathbf{B}\mathbf{u}^{(n)}, Q) \end{aligned} \quad (9.85)$$

Next, we find the log of the expression in Eq. (9.84). This is our complete log-likelihood:

$$\begin{aligned} \log p(\{\mathbf{x}\}_0^N, \{\mathbf{y}\}_1^N | \theta, \{\mathbf{u}\}_1^N) &= -\sum_{n=1}^N \frac{1}{2} (\mathbf{y}^{(n)} - \mathbf{C}\mathbf{x}^{(n)})^T R^{-1} (\mathbf{y}^{(n)} - \mathbf{C}\mathbf{x}^{(n)}) \\ &\quad - \sum_{n=1}^N \frac{1}{2} (\mathbf{x}^{(n)} - \mathbf{A}\mathbf{x}^{(n-1)} - \mathbf{B}\mathbf{u}^{(n-1)})^T Q^{-1} (\mathbf{x}^{(n)} - \mathbf{A}\mathbf{x}^{(n-1)} - \mathbf{B}\mathbf{u}^{(n-1)}) \\ &\quad - \frac{1}{2} (\mathbf{x}^{(0)} - \hat{\mathbf{x}}^{(0)})^T P_0^{-1} (\mathbf{x}^{(0)} - \hat{\mathbf{x}}^{(0)}) \\ &\quad - \frac{N}{2} \log |R| - \frac{N}{2} \log |Q| - \frac{1}{2} \log |P_0| + \text{const} \end{aligned} \quad (9.86)$$

In the E step, we fix the parameters θ and find the state estimate $\{\hat{\mathbf{x}}\}_1^N$ that maximizes the expected value of the quantity in Eq. (9.86). The state estimate is typically the Kalman estimate, or better yet the estimate that depends both on the past and future observations of \mathbf{y} : this is called smoothing (Anderson and Moore, 1979), in which

$$\hat{\mathbf{x}}^{(n)} \equiv E[\mathbf{x}^{(n)} | \{\mathbf{y}\}_1^N] \quad (9.87)$$

In the M step, we fix the state estimate $\{\hat{\mathbf{x}}\}_1^N$, and find the parameters θ that maximize the complete log-likelihood. To do so, we find the derivative of the expected value of the quantity in Eq. (9.86) with respect to parameters θ , set it to zero, and then use the derivative to find the estimate of these parameters. To show how to do this, let us label the complete log-likelihood expression in Eq. (9.86) by the short hand l_c . We have:

$$\begin{aligned} \frac{dl_c}{dC} &= \sum_{n=1}^N R^{-1} \mathbf{y}^{(n)} \mathbf{x}^{(n)T} - R^{-1} \mathbf{C} \sum_{n=1}^N \mathbf{x}^{(n)} \mathbf{x}^{(n)T} = 0 \\ E\left[\frac{dl_c}{dC}\right] &= \sum_{n=1}^N R^{-1} \mathbf{y}^{(n)} \hat{\mathbf{x}}^{(n)T} - R^{-1} \mathbf{C} P^{(n)} = 0 \end{aligned} \quad (9.88)$$

In Eq. (9.88), the matrix $P^{(n)}$ refers to the variance of the state estimate,

$P^{(n)} \equiv E[\mathbf{x}^{(n)} \mathbf{x}^{(n)T} | \{\mathbf{y}\}_1^N]$. Solving for C , we have:

$$C_{new} = \left[\sum_{n=1}^N \mathbf{y}^{(n)} \hat{\mathbf{x}}^{(n)T} \right] \left[\sum_{n=1}^N P^{(n)} \right]^{-1} \quad (9.89)$$

To find a new estimate for R , it is convenient to find the derivative of the log-likelihood with respect to R^{-1} :

$$\frac{dl_c}{dR^{-1}} = \frac{N}{2} R + \sum_{n=1}^N C \mathbf{x}^{(n)} \mathbf{y}^{(n)T} - \frac{1}{2} C \mathbf{x}^{(n)} \mathbf{x}^{(n)T} C^T - \frac{1}{2} \mathbf{y}^{(n)} \mathbf{y}^{(n)T} \quad (9.90)$$

Setting the above quantity to zero gives us the new estimate for R :

$$R_{new} = \frac{1}{N} \left(\sum_{n=1}^N C P^{(n)} C^T + \mathbf{y}^{(n)} \mathbf{y}^{(n)T} - 2C \hat{\mathbf{x}}^{(n)} \mathbf{y}^{(n)T} \right) \quad (9.91)$$

For parameter A we have:

$$\frac{dl_c}{dA} = \sum_{n=1}^N Q^{-1} \mathbf{x}^{(n)} \mathbf{x}^{(n-1)T} - Q^{-1} A \mathbf{x}^{(n-1)} \mathbf{x}^{(n-1)T} - Q^{-1} B \mathbf{u}^{(n-1)} \mathbf{x}^{(n-1)T} \quad (9.92)$$

The new estimate for A becomes:

$$A_{new} = \left(\sum_{n=1}^N P^{(n,n-1)} - B \mathbf{u}^{(n-1)} \hat{\mathbf{x}}^{(n-1)T} \right) \left(\sum_{n=1}^N P^{(n-1)} \right)^{-1} \quad (9.93)$$

For parameter B we have:

$$\frac{dl_c}{dB} = \sum_{n=1}^N Q^{-1} \mathbf{x}^{(n)} \mathbf{u}^{(n-1)T} - Q^{-1} A \mathbf{x}^{(n-1)} \mathbf{u}^{(n-1)T} - Q^{-1} B \mathbf{u}^{(n-1)} \mathbf{u}^{(n-1)T} \quad (9.94)$$

The new estimate for B becomes:

$$B_{new} = \left(\sum_{n=1}^N \hat{\mathbf{x}}^{(n)} \mathbf{u}^{(n-1)T} - A \hat{\mathbf{x}}^{(n-1)} \mathbf{u}^{(n-1)T} \right) \left(\sum_{n=1}^N \mathbf{u}^{(n-1)} \mathbf{u}^{(n-1)T} \right)^{-1} \quad (9.95)$$

Finally, for parameter Q we have:

$$\begin{aligned} \frac{dl_c}{dQ^{-1}} = & \frac{N}{2} Q + \sum_{n=1}^N \mathbf{x}^{(n)} \mathbf{x}^{(n)T} + A \mathbf{x}^{(n-1)} \mathbf{x}^{(n)T} + B \mathbf{u}^{(n-1)} \mathbf{x}^{(n)T} \\ & - A \mathbf{x}^{(n-1)} \mathbf{x}^{(n-1)T} A^T - A \mathbf{x}^{(n-1)} \mathbf{u}^{(n-1)T} B^T - \frac{1}{2} B \mathbf{u}^{(n-1)} \mathbf{u}^{(n-1)T} B^T \end{aligned} \quad (9.96)$$

The new estimate for Q becomes:

$$\begin{aligned} Q_{new} = & \frac{1}{N} \sum_{n=1}^N P^{(n)} - 2A P^{(n-1,n)} - 2B \mathbf{u}^{(n-1)} \hat{\mathbf{x}}^{(n)T} + 2A P^{(n-1)} A^T \\ & + 2A \hat{\mathbf{x}}^{(n-1)} \mathbf{u}^{(n-1)T} B^T + B \mathbf{u}^{(n-1)} \mathbf{u}^{(n-1)T} B^T \end{aligned} \quad (9.97)$$

Summary

In building a model that can predict things, one can begin with some generic structure and then optimally fit the parameters of this structure to the observed data. This is the problem of state

estimation, something that we considered in the last few chapters. However, a generic structure often has the disadvantage of being poorly suited to the task at hand, resulting in poor generalization, and slow learning. It would be useful to build a new structure or model topology that is specific to the data that one observes. This would give one the capability to learn to ride a small bike at childhood, and then generalize to larger bikes as one grows older.

The mathematical problem is one of finding the structure of a stochastic dynamical system that can, in principle, be given a sequence of inputs and produce the sequence of outputs that match that observed from the real system. For linear stochastic systems, a closed form solution exists and is called subspace analysis. This approach relies on the fact that each observation $\mathbf{y}^{(n)}$ is a linear combination of states $\mathbf{x}^{(n)}$ and inputs $\mathbf{u}^{(n)}$, which means that vectors $\mathbf{x}^{(n)}$ and $\mathbf{u}^{(n)}$ are the bases for vector $\mathbf{y}^{(n)}$. By projecting the vector $\mathbf{y}^{(n)}$ onto a vector perpendicular to $\mathbf{u}^{(n)}$, one is left with a vector in the subspace spanned by $\mathbf{x}^{(n)}$ and is proportional to $\mathbf{x}^{(n)}$. Recovery of this subspace in which the hidden states $\mathbf{x}^{(n)}$ reside allows one to find the structure of the dynamical system.

An application of this formalism is in modeling biological system and how they learn. The idea is to give some inputs to the learner, and observe their behavior, and then use that input-output relationship to discover the structure of the learner. An example of this is in saccade adaptation, in which it was found that in a typical single session experiment, the learner is well represented via a fast system that decays with a time constant of about 25 seconds, and is highly sensitive to errors, and a slow system that shows very little if any decay, and is also an order of magnitude less sensitive to error.

An alternate approach to estimating the structure of a linear dynamical system is via Expectation Maximization. In this procedure, one begins by assuming a structure and then finds the hidden states. Next, one uses these estimates of the hidden states and finds the structural quantities.

Figure Legends

Figure 9.1. A schematic of an arm moving in the horizontal plane with and without a tennis racquet. The circled plus signs indicate center of mass of each link of the arm.

Figure 9.2. Subjects trained in a long sequence of trials in which a rotation was imposed on the motion of a cursor. The trial-to-trial distribution of the rotation perturbation was random. After this period of training, performance was measured in a sequence of trials in which the perturbation was a constant $+60^\circ$ rotation. Their data is shown in the above plot in the Random rotation group. Performance was significantly better than a naïve group who had prior training in trials for which there were no perturbations. Performance was also better than another group in which prior training was in a perturbation that had rotation, shearing, and scaling (labeled as Random linear transform). (Figure from (Braun et al., 2009), with permission.)

Figure 9.3. Projecting a vector onto the subspace spanned by the row vectors of a matrix. **A)** When we project matrix A onto B , we are projecting the row vectors of A (Eq. 9.18) onto the subspace spanned by the row vectors of B (Eq. 9.19). **B)** We use the term B^\perp to define the space that is perpendicular to the space spanned by row vectors of B . Therefore, B^\perp is simply a line in this example.

Figure 9.4. Dynamics of some sample systems. **A)** System of Eq. (9.57), driven by inputs $\{u^{(1)}, u^{(2)}, \dots, u^{(p)}\}$ that are shown and producing outputs $\{y^{(1)}, y^{(2)}, \dots, y^{(p)}\}$. **B)** System of Eq. (9.59) driven by inputs that are shown and producing the plotted outputs. **C)** System of Eq. (9.62) driven by random noise inputs that are shown and producing the plotted outputs.

Figure 9.5. Identifying the structure of learners in a saccade adaptation experiment. **A)** Experimental protocol. The experiment began with error-clamp trials, was followed by a gain-down session, followed by a gain-up session (‘extinction’), and finally error-clamp trials. **B)** The averaged saccade amplitudes produced by a group of subjects. **C)** Model result. Subspace analysis was used to identify the structure of the system, and then a Kalman filter was used to give a running estimate of the two hidden states. **D)** Estimate of the two hidden states. **E)** The time constants of the two states (forgetting rates), the error sensitivity of the two states, and the

contribution of the two states to output. The values for fits to individual subjects are shown, as is the mean of that distribution. (From (Ethier et al., 2008), with permission.)

Reference List

- Anderson BDO, Moore JB (1979) *Optimal Filtering*. Englewood Cliffs, N.J.: Prentice-Hall.
- Braun DA, Aertsen A, Wolpert DM, Mehring C (2009) Motor task variation induces structural learning. *Curr Biol* 19:352-357.
- Cheng S, Sabes PN (2006) Modeling sensorimotor learning with linear dynamical systems. *Neural Comput* 18:760-793.
- Ethier V, Zee DS, Shadmehr R (2008) Spontaneous recovery of motor memory during saccade adaptation. *J Neurophysiol* 99:2577-2583.
- Ghahramani Z, Hinton GE (1996) Parameter estimation for linear dynamical systems. In: Technical Report CRG-TR-96-2 Univ. Toronto.
- Shumway RH, Stoffer DS (1982) An approach to time series smoothing and forecasting using the EM algorithm. *J Time Series Analysis* 3:253-264.
- van Overschee P, De Moor B (1996) *Subspace identification for linear systems*. Boston: Kluwer Academic.
- Vaziri S, Diedrichsen J, Shadmehr R (2006) Why does the brain predict sensory consequences of oculomotor commands? Optimal integration of the predicted and the actual sensory feedback. *J Neurosci* 26:4188-4197.