Lecture 26
Theory of Dynamic Mode Decomposition (DMD)

Jason J. Bramburger

You're probably noticing a bit of a pattern that comes from using SVD/PCA/POD. We REALLY care about dimensionality reduction. And this is to be expected. The world is full of high-dimensional datasets, but many of them use tons of different observable variables to describe something that when viewed the right way turns out to be low-dimensional. What we saw in the previous two lectures with POD reductions was called a *model-based* algorithm. This is because we had the governing equations - the model. So, what are we supposed to do if we just collect data and don't know or don't have access to complete governing equations? This leads to the notion of *data-based* algorithms.

In this lecture we are going to focus on a relatively new technique called **Dynamic Mode Decomposition** (DMD). The aim of the method is to take advantage of low-dimensionality in experimental data without having to rely on a given set of governing equations. We will see that DMD will allow us to *forecast* our data, that is, predict what it will be in the future. This is different from what we did with POD in Lecture 24 since POD gives orthogonal basis functions to optimally describe your data, but it doesn't tell you how they will evolve in time - just how they were evolving on the training set. DMD is impressive since it finds a basis of spatial modes (not necessarily orthogonal) for which the time dynamics are just exponential functions (with potentially complex exponents). Hence, the only thing that happens is time is oscillations, decay, and/or growth. This should remind you of linear systems of differential equations, because that's essentially what DMD turns the problem into!

The Setup

We are going to assume that we have *snapshots* of spatio-temporal data. That is, data that evolves in both space and time. The 'space' part can be loosely defined since we really just need a collection of vectors that all evolve in time. We will define:

$$N = \text{number of spatial points saved per unit time snapshot}$$
$$M = \text{number of snapshots taken.}$$

The most import thing is that the time data is collected at regularly spaced intervals:

$$t_{m+1} = t_m + \Delta t, \qquad m = 1, \dots, M-1, \qquad \Delta t > 0.$$

The snapshots are denoted

$$U(\mathbf{x}, t_m) = \begin{bmatrix} U(x_1, t_m) \\ U(x_2, t_m) \\ \vdots \\ U(x_n, t_m) \end{bmatrix}$$

for each $m = 1, \dots, M$. We can use these snapshots to form columns of data matrices

$$\mathbf{X} = \begin{bmatrix} U(\mathbf{x}, t_1) & U(\mathbf{x}, t_2) & \cdots & U(\mathbf{x}, t_M) \end{bmatrix},$$

and

$$\mathbf{X}_j^k = \begin{bmatrix} U(\mathbf{x}, t_j) & U(\mathbf{x}, t_{j+1}) & \cdots & U(\mathbf{x}, t_k) \end{bmatrix}.$$

The matrix $\mathbf{X}_j^k$ is just columns j through k of the full snapshot matrix $\mathbf{X}$.

The Koopman Operator

The DMD method approximates the modes of the **Koopman operator**. Here's the general idea. The Koopman operator $\mathbf{A}$ is a linear, time-independent operator such that

$$\mathbf{x}_{j+1} = \mathbf{A}\mathbf{x}_j,$$

where the j indicates the specific data collection time and $\mathbf{A}$ is the linear operator that maps the data from time t_j to t_{j+1}. The vector $\mathbf{x}_j$ is an N-dimensional vector of the data points collected at time j. That is, applying $\mathbf{A}$ to a snapshot of data will advance it forward in time by Δt. This is crazy if you think about it! We are asking for a linear mapping from one timestep to the next, even though the dynamics of the system are likely nonlinear.

Note: You might have learned about linearizing ODEs to understanding *local* properties of the dynamics. This is different. The Koopman operator is *global* - it's not restricted to an area of space or time.

Dynamic Mode Decomposition

To construct the appropriate Koopman operator that best represents the data collected, we will consider the matrix

$$\mathbf{X}_1^{M-1} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_2 & \cdots & \mathbf{x}_{M-1} \end{bmatrix},$$

where we use the shorthand $\mathbf{x}_j$ to denote a snapshot of the data at time t_j. The Koopman operator allows us to rewrite this as

$$\mathbf{X}_1^{M-1} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{A}\mathbf{x}_1 & \mathbf{A}^2\mathbf{x}_1 & \cdots & \mathbf{A}^{M-2}\mathbf{x}_1 \end{bmatrix}.$$

The columns are formed by applying powers of $\mathbf{A}$ to the vector $\mathbf{x}_1$, and are said to form the basis for the Krylov subspace. Hence, we have (in theory at least) a way of relating the first $M-1$ snapshots to $\mathbf{x}_1$ using just the Koopman operator/matrix.

We can write the above in matrix form to get

$$\mathbf{X}_2^M = \mathbf{A}\mathbf{X}_1^{M-1} + \mathbf{r}e_{M-1}^T,$$

where e_{M-1} is the vector with all zeros except a 1 at the $(M-1)$st component. That is, $\mathbf{A}$ applied to each column of $\mathbf{X}_1^{M-1}$, given by $\mathbf{x}_j$, maps to the corresponding column of $\mathbf{X}_2^M$, given by $\mathbf{x}_{j+1}$, but the final point $\mathbf{x}_M$ wasn't included in our Krylov basis, so we add in the *residual* (or error) vector $\mathbf{r}$ to account for this. Remember that $\mathbf{A}$ is unknown and it is our goal to find it. We should also remember that matrices are completely understood by their eigenvalues and eigenvectors, so we are going to circumvent finding $\mathbf{A}$ directly by finding another matrices with the same eigenvalues. We will see that the eigenvectors come easily from there.

First, let's use the SVD to write $\mathbf{X}_1^{M-1} = \mathbf{U}\Sigma\mathbf{V}^*$. Then, from above we get

$$\mathbf{X}_2^M = \mathbf{A}\mathbf{U}\Sigma\mathbf{V}^* + \mathbf{r}e_{M-1}^T.$$

We are going to choose $\mathbf{A}$ in such a way that the columns in $\mathbf{X}_2^M$ can be written as linear combinations of the columns of $\mathbf{U}$. This is the same as requiring that they can be written as linear combinations of the POD modes. Hence, the residual vector $\mathbf{r}$ must be orthogonal to the POD basis, giving that $\mathbf{U}^*\mathbf{r} = 0$. Multiplying the above equation through by $\mathbf{U}^*$ on the left gives:

$$\mathbf{U}^*\mathbf{X}_2^M = \mathbf{U}^*\mathbf{A}\mathbf{U}\Sigma\mathbf{V}^*.$$

Then, we can isolate for $\mathbf{U}^*\mathbf{A}\mathbf{U}$ by multiplying by $\mathbf{V}$ and then Σ^{-1} on the right to get

$$\mathbf{U}^*\mathbf{A}\mathbf{U} = \underbrace{\mathbf{U}^*\mathbf{X}_2^M\mathbf{V}\Sigma^{-1}}_{=:\tilde{\mathbf{S}}}.$$

Remember, everything on the right side is known from the input data, and I'm writing the tilde over the $\mathbf{S}$ to keep with convention in the literature.

Important Note: The goal a lot of the time with these data-driven method is dimensionality reduction. Therefore, we don't have to use the full matrix Σ, we can use Σ with just the first $K \geq 1$ nonzero singular values on the diagonal. From what we know about singular values, K is the rank of the data matrix $\mathbf{X}_1^{M-1}$, which essentially tells us the number of dimensions that the data varies in. Ideally we would like K to be small relative to N and M. This also explains why I just wrote Σ^{-1} instead of the pseudoinvers above - I'm assuming Σ is invertible because we got rid of the singular values that are zero. To be perfectly precise, this means that

$$\mathbf{U} \in \mathbb{C}^{N\times K}, \quad \Sigma \in \mathbb{R}^{K\times K}, \quad \mathbf{V} \in \mathbb{C}^{M-1\times K}.$$

Notice that $\tilde{\mathbf{S}}$ and $\mathbf{A}$ are related by applying a matrix on one side and its inverse on the other. This means they are *similar*. Similar matrices share a lot of properties, including that they have the same eigenvalues! Furthermore, if $\mathbf{y}$ is an eigenvector of $\tilde{\mathbf{S}}$, then $\mathbf{U}y$ is an eigenvector of $\mathbf{A}$. So, let's write the eigenvector/eigenvalue pairs of $\tilde{\mathbf{S}}$ as

$$\tilde{\mathbf{S}}\mathbf{y}_k = \mu_k \mathbf{y}_k,$$

thus giving the eigenvectors of $\mathbf{A}$, called the *DMD modes*, by

$$\psi_k = \mathbf{U}\mathbf{y}_k.$$

Now we've got all we need to describe continual multiplications by $\mathbf{A}$! We can just expand in our eigenbasis to get

$$\mathbf{x}_{DMD}(t) = \sum_{k=1}^{K} b_k \psi_k \mathrm{e}^{\omega_k t} = \Psi \mathrm{diag}(\mathrm{e}^{\omega_k t})\mathbf{b}.$$

As discussed in the important note above, K is the rank of $\mathbf{X}_1^{M-1}$. The b_k are the initial amplitude of each mode, and the matrix Ψ contains the eigenvectors ψ_k as its columns. I am using the convection from ODEs that we write the time dynamics as exponential functions, meaning that $\omega_k = \ln(\mu_k)/\Delta t$. This convention is also why it looks like I seemingly jumped to continuous time above - technically the t can only move forward in discrete values by steps of size Δt, but in the real world time is continuous and so we can interpolate between time jumps with the above formula.

The above is nothing more than an eigenvalue expansion you might have seen working with ODEs. We just take the linear dynamics governed by $\mathbf{A}$, compute the eigenvalues and eigenvectors, and then write the dynamics in terms of a linear combination of exponential growth/decay/oscillation in the direction of each eigenvector. So, how do we compute the b_k? Easy! We know that at time $t = 0$ in the above formula we have to get $\mathbf{x}_1$ because this was our initial condition to generate the other $\mathbf{x}_m$. Therefore, taking $t = 0$ in the above gives

$$\mathbf{x}_1 = \Psi\mathbf{b} \implies \mathbf{b} = \Psi^{\dagger}\mathbf{x}_1,$$

where $\Psi^{\dagger}$ is the pseudoinverse of the matrix Ψ.

Summary and Algorithm

The DMD algorithm takes advantage of low dimensionality in the data to create a low-rank approximation of the linear mapping that best iterates you through the snapshots of the data. Think about how cool this is: no one ever said the snapshots have to be governed by something linear! In the next lecture we will put this into practice using the NLS again, which is certainly a nonlinear system. The really exciting thing about DMD is that it is completely *data-driven*. It doesn't require governing equations! You just need data snapshots to apply the algorithm.

Here is an annotated summary of the algorithm:

1. Sample data at N prescribed locations M times. The data snapshots should be evenly spaced in time by a fixed Δt. Use the snapshots to create the matrix $\mathbf{X}$.
2. From the data matrix $\mathbf{X}$, construct the two submatrices $\mathbf{X}_1^{M-1}$ and $\mathbf{X}_2^{M}$.
3. Compute the SVD of $\mathbf{X}_1^{M-1}$.
4. Create the matrix $\tilde{\mathbf{S}} = \mathbf{U}^*\mathbf{X}_2^M\mathbf{V}\Sigma^{-1}$ and find its eigenvalues and eigenvectors.
5. Use the initial snapshot $\mathbf{x}_1$ and the pseudoinverse of Ψ to find the coefficients b_k.
6. Compute the solution at any future time using the DMD modes along with their projection to the initial conditions and the time dynamics computed using the eigenvalues of $\tilde{\mathbf{S}}$.

Bonus: You don't have to use your first snapshot as the initial condition. You could use anything! It might not work as well since you 'trained' on the snapshots, but it's worth a try!