Bruxelles, November 2018

Notes on Dynamic Mode Decomposition
(with some code)

camillejr.github.io/science-docs

Preface

Dynamic Mode Decomposition (DMD) is a data-driven method of finding low-rank structures in high-dimensional data sets.
These notes are taken from two lectures on Dynamic Mode Decomposition: [1] and [2] by Prof. Nathan Kutz from the University of Washington.

This document is still in preparation. Please feel free to contact me with any suggestions, corrections or comments.

Contents

1 Description of the system

We have a system described by a differential equation:

$$\frac{d\vec{x}}{dt} = f(\vec{x}, t, \mu) \tag{1}$$

The function $f(\vec{x}, t, \mu)$ is a way of *modeling* that system. We also have *measurements* of the system in different points in space at time k, in the form of a vector(s) $\vec{y}_k$:

$$\vec{y}_k = g(\vec{x}_k) \tag{2}$$

where $\vec{x}_k$ is the quantity of interest that we are aiming at measuring. The fact that we might not be able to measure it directly is accounted for by some function $g()$ (although it might happen that $\vec{y}_k = \vec{x}_k$, meaning that we are able to measure $\vec{x}_k$ directly).
Notice, that for measurements at many moments in time, we may stack all the collected vectors $\vec{y}_i$ for different times i to create a matrix whose columns represent time snapshots and whose rows represent position in space.

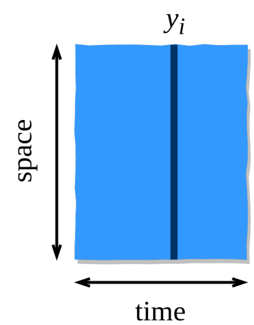

Figure 1: Data matrix with measurements of the system.

2 Linear dynamical systems

We are from now interested in systems where the governing equation from eq.(1) is not known (in other words, the function f is unknown) and we solely rely on measurements of the system which, in general, form a high-dimensional data set.
In the Dynamic Mode Decomposition we approximate that data set by a linear dynamical system of the form:

$$\frac{d\vec{x}}{dt} = \boldsymbol{A}\vec{x} \tag{3}$$

This is in fact a very handy approximation since we are able to write down exact solutions to linear systems.
Once we assume that the general solution is of the form:

$$\vec{x} = \vec{v}e^{\boldsymbol{\lambda} t} \tag{4}$$

to obtain the parameters we effectively solve the eigenvalue problem:

$$\boldsymbol{A}\vec{v} = \boldsymbol{\lambda}\vec{v} \tag{5}$$

The exact solution to the linear system from eq.(3) is:

$$x = \sum_{j=1}^{n} b_j \phi_j e^{\lambda_j t} \tag{6}$$

For a reader who is now shaky about how this solution was derived, more can be found in appendix A.

3 Dynamic Mode Decomposition theory

3.1 Exact DMD

For the moment, we assume that we can measure the system directly, that is we measure $\vec{y}_i = \vec{x}_i$. Moreover, we assume that our data is collected in equal[1] time steps Δt. The measurements are combined inside a large matrix $\boldsymbol{X}$ where each of its columns represents one time snapshot:

$$\boldsymbol{X} = \begin{bmatrix} \vec{x}_1 & \vec{x}_2 & \vec{x}_3 & \dots & \vec{x}_m \end{bmatrix} \tag{7}$$

We split the large matrix $\boldsymbol{X}$ into two matrices $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$ such that:

$$\boldsymbol{X_1} = \begin{bmatrix} \vec{x}_1 & \vec{x}_2 & \vec{x}_3 & \dots & \vec{x}_{m-1} \end{bmatrix} \tag{8}$$

$$\boldsymbol{X_2} = \begin{bmatrix} \vec{x}_2 & \vec{x}_3 & \vec{x}_4 & \dots & \vec{x}_m \end{bmatrix} \tag{9}$$

If we now assume that a linear operator will map the first element of $\boldsymbol{X_1}$ with the first element of $\boldsymbol{X_2}$, second with the second, third with the third, and so on, matrix $\boldsymbol{X_2}$ can be thought of as a matrix representing the *future state* of the matrix $\boldsymbol{X_1}$. That linear operator is assumed to be a matrix $\boldsymbol{A}$.

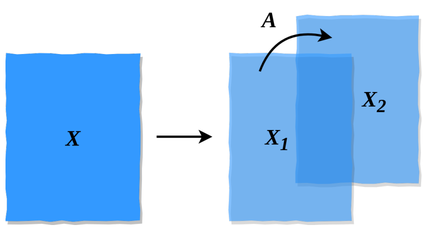

Figure 2: Spliting the data matrix into *past* and *future* matrices $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$, linked by the linear operator $\boldsymbol{A}$.

Note here, that for nonlinear systems, a matrix that transforms $\vec{x}_1$ to $\vec{x}_2$ is different from a matrix that transforms $\vec{x}_2$ to $\vec{x}_3$ and so on. DMD assumes, however, that there is one matrix $\boldsymbol{A}$ that does all these transformations at once, with the least amount of error. It finds the *best-fit* linear dynamical system for the non-linear data set. In mathematical terms, we are looking for such $\boldsymbol{A}$ that:

$$\boldsymbol{X_2} = \boldsymbol{A}\boldsymbol{X_1} \tag{10}$$

To solve such system we multiply both sides by the *pseudo-inverse* of matrix $\boldsymbol{X_1}$ which we denote by $\boldsymbol{X_1^+}$:

$$\boldsymbol{A} = \boldsymbol{X_2}\boldsymbol{X_1^+} \tag{11}$$

The pseudo-inverse described here, also known as the Moore-Penrose inverse[2], is computed using the least squares method. There is therefore certain information lost when going from eq.(10) to eq.(11).

Once we have solved for matrix $\boldsymbol{A}$, we can go back to eq.(5) and solve for eigenvalues and eigenvectors.

Up to this point, this is what the **exact DMD** computes. There is however a problem that the eq.(11) may pose when numerics are involved and this will be addressed in the next section.

3.2 Going low-rank

Matrices $\boldsymbol{X_1^+}$ and $\boldsymbol{X_2}$ typically represent huge spacial dimensionality[3] which in turn means that the matrix $\boldsymbol{A}$ can become a square matrix of a massive size.

We are hence reluctant to perform the multiplication of matrices as is stated in eq.(11).

The hope comes from the *Singular Value Decomposition* (SVD). We belive that there are low-rank structures hidden in the data set and we are able to reduce the dimensionality of matrix $\boldsymbol{A}$ without significant loss of information[4].

We perform the SVD on matrix $\boldsymbol{X_1}$:

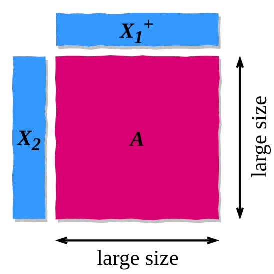

Figure 3: Building the linear operator $\boldsymbol{A}$ in exact DMD.

$$\boldsymbol{X_1} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \tag{12}$$

Based on the rank structure of the matrix $\boldsymbol{X_1}$ (one way to get information about the rank structure is to plot the elements from the diagonal of the matrix $\boldsymbol{\Sigma}$) we perform a rank-r truncation on the SVD decomposition and approximate the matrix $\boldsymbol{X_1}$ by its low-rank (rank-r) representation:

[1] Which is indeed a special case for real life measurements. Check section 4 for more information.

[2] Check appendix B for more information.

[3] This is often the case for data sets where we have very few snapshots in time but a large number of spacial points where the measurements were taken. Graphically, we might think of those matrices as being "tall" and this is illustrated in Figure 1.

[4] Professor Kutz said a very interesting sentence here, that the multiplication presented in Figure 3 completely ignores the fact that there might be low-rank structures in our data set.

$$\boldsymbol{X_1} \approx \boldsymbol{X_{1r}} = \boldsymbol{U_r \Sigma_r V_r^T} \tag{13}$$

The pseudo-inverse of the truncated matrix is:

$$\boldsymbol{X_{1r}^+} = \boldsymbol{V_r \Sigma_r^{-1} U_r^T} \tag{14}$$

The usefulness of this decomposition might not yet be evident, since the matrix $\boldsymbol{X_{1r}}$ is of the same size as matrix $\boldsymbol{X_1}$, they only differ by rank. The idea is to nevertheless use the SVD decomposition but also, to generate a matrix similar to the matrix $\boldsymbol{A}$ (since similar matrices share eigenvalues and eigenvectors, among some other properties) but one that will have a smaller size (in fact, it will be size $(r \times r)$). This similar matrix will be denoted $\underline{\boldsymbol{A}}$. Since it has a lower size than the original matrix $\boldsymbol{A}$, we will only retrieve r eigenvectors and eigenvalues.

What will now follow are clever mathematical steps performed to avoid computation of the large matrix $\boldsymbol{A}$.

We come back to the eq.(11) and

We perform a *similarity transform* of the matrix $\boldsymbol{A}$:

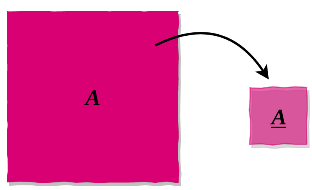

Figure 4: Similarity transform of matrix $\boldsymbol{A}$ to reduce the size.

$$\underline{\boldsymbol{A}} = \boldsymbol{U_r^T A U_r} \tag{15}$$

Matrix $\boldsymbol{A}$ can be written as:

$$\boldsymbol{A} = \boldsymbol{X_2 V_r \Sigma_r^{-1} U_r^T} \tag{16}$$

The similar matrix $\underline{\boldsymbol{A}}$ can be written as:

$$\underline{\boldsymbol{A}} = \boldsymbol{U_r^T X_2 V_r \Sigma_r^{-1}} \tag{17}$$

taking into account that $\boldsymbol{U_r^T U_r} = \boldsymbol{I}$.

We have thus chosen a low-dimensional subspace by performing rank-r truncation in which we now find the solution to the linear dynamical system presented initially. The model for the solution is built in this low-dimensional subspace.

3.3 Eigendecomposition

Now that we have computed the similar matrix $\underline{\boldsymbol{A}}$, we move on to perform the eigendecomposition:

$$[\boldsymbol{W}, \boldsymbol{\Lambda}] = \text{eig}(\underline{\boldsymbol{A}}) \tag{18}$$

3.4 Going back to the original dimensions

Once the model has been built in the low-dimensional subspace, we want to move to the original dimensions. The DMD modes are obtained from:

$$\boldsymbol{\Phi} = \boldsymbol{X_2 V \Sigma^{-1} W} \tag{19}$$

DMD modes are not orthogonal. This creates a great capacity of DMD to be applicable to systems where data structure does not exhibit orthogonality.

The solution to the original dynamical system is finally computed:

$$\vec{x}(t) = \boldsymbol{\Phi} e^{\boldsymbol{\Omega} t} \vec{b} \tag{20}$$

the above equation is equivalent to:

$$\vec{x}(t) = \sum_{k=1}^{r} \phi_k e^{\omega_k t} b_k \tag{21}$$

4 A broader view on DMD

What can go wrong with our data sets?

4.1 Optimized DMD

- varying time steps

We mentioned earlier, that

4.2 Robust DMD

Sparse Identification

5 Python example

A Solution to linear dynamical systems

We first recall the general solution to the differential equation:

$$\frac{df(x)}{dt} = f(x) \tag{22}$$

to be the exponential function: $f(x) = a \cdot e^x$.

In an analogous way, the general solution to the linear dynamical system of the form:

$$\frac{d\vec{x}}{dt} = \boldsymbol{A}\vec{x} \tag{23}$$

is:

$$\vec{x} = \vec{v} e^{\boldsymbol{\lambda} t} \tag{24}$$

Computing the time derivative of the eq. 24 we get:

$$\frac{d\vec{x}}{dt} = \vec{v}\boldsymbol{\lambda}e^{\boldsymbol{\lambda}t} \tag{25}$$

And substituting the eq. 24 to eq. 23 we get:

$$\frac{d\vec{x}}{dt} = \boldsymbol{A}\vec{v}e^{\boldsymbol{\lambda}t} \tag{26}$$

The nontrivial solution for the equality of these two above equations is obtained when:

$$\boldsymbol{A}\vec{v} = \boldsymbol{\lambda}\vec{v} \tag{27}$$

which is the statement of eigenvalue problem.

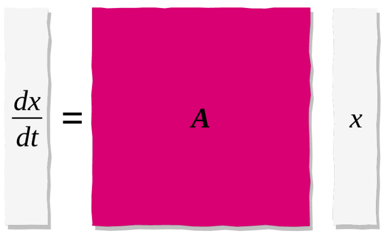

Figure 5: Linear dynamical system.

B Moore-Penrose inverse

C Singular Value Decomposition

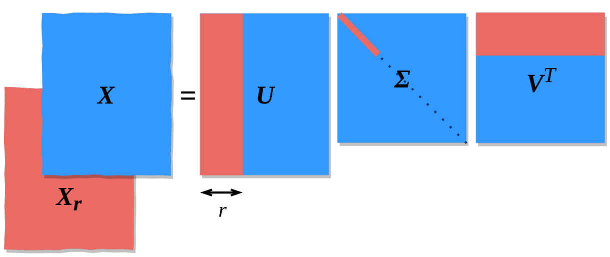

Figure 6: Sizes of component matrices in the Singular Value Decomposition and after rank truncation.

References

[1] N. Kutz, *Dynamic Mode Decomposition Theory*, an online lecture: https://youtu.be/bYfGVQ1Sg98

[2] N. Kutz, *Dynamic Mode Decomposition Code* , an online lecture: https://youtu.be/KAau5TBU0Sc

[3] E. R. Scheinerman, *Invitation to Dynamical Systems*

[4] G. Strang, *Introduction to Linear Algebra*, 5th edition

[5] K. Zdybal, The von Karman Institute for Fluid Dynamics: *POD and DMD decomposition of numerical and experimental data*, stagiaire report