AMATH 482 Homework 5

Jiaqi Su

March 16, 2021

Abstract

Videos in real life can be divided into frames and therefore can be regarded as a sets of spatial-temporal data. In this way, the Dynamic Mode Decomposition can be used to find spatial modes and create low-rank approximation of our data. Separating our data matrix into a low rank approximation and the remaining sparse one, we can separate our video into backgrounds and foregrounds.

1 Introduction and Overview

The expectation of this problem is to separate the given videos to a background video and a foreground video. To achieve this, a proper DMD (Dynamic modes decomposition) reconstruction of our image frames in the video can be used to divide the image frames into two background and foreground based on whether the part changes through time.

2 Theoretical Background

2.1 Dynamic Modes Decomposition

Given a sets of screenshots of video which evolves in both space and time, assuming we have N spatial points and M time points, we can create a matrix to represent all of data such that

$$X = [U(x, t_1), U(x, t_2), ..., U(x, t_m)] \tag{1}$$

where $U(x, t_k)$ represents the all spatial data at time k. And we use

$$X_j^k = [U(x, t_j), U(x, t_j + 1), ..., U(x, t_k)] \tag{2}$$

to denote data from the j th screenshot to the k th screenshot.
To approximate the modes of the data, DMD treats the data as a time-advancing linear system with a matrix operator A:

$$x_{j+1} = Ax_j \tag{3}$$

,in which we can have a linear mapping from time j to time $j + 1$.
Applying the idea of Koopman operator into our data matrix X, we can relate our first M-1 frames of the videos to x_1:

$$X_1^{M-1} = [x_1, Ax_1, A^2x_1, ..., A^{M-2}x_1] \tag{4}$$

and therefore we can derive

$$X_2^M = AX_1^{M-1} + re_{M-1}^T \tag{5}$$

where the residual vector r accounts for excluding x_M in our Krylov basis formed by columns of X_1^{M-1}.
To find modes of our Koopman operators, we can find another matrix similar to A and thus with the same eigenvalues. Using svd we have $X_1^{M-1} = U\Sigma V^*$, and therefore replace the X_1^{M-1} in last equation we get:

$$X_2^M = AU\Sigma V^* + re_{M-1}^T \tag{6}$$

We will choose A in a way that the columns of X_2^M are a linear combination of columns of U, and therefore provide a basis orthogonal to our residual vector r, giving $U^*r = 0$. So multiplying U^* gives us:

$$U^*X_2^M = U^*AU\Sigma V^* \tag{7}$$

As we said the modes are eigenvectors of A, so we can separate U^*AU such that

$$U^*AU = U^*X_2^M V\Sigma^{-1} \tag{8}$$

Denote the right hand-side as $\tilde{S}$, this can be calculated using our data and then used to find the eigenvalues and eigenvectors of A since the matrix U^*AU on the left hand side is similar to A. Therefore, if we have y_k is the k th eigenvector of $\tilde{S}$, the eigenvectors of A will just be

$$\phi_k = Uy_k \tag{9}$$

Expand in the eigenbasis, we can have the k-rank approximation of the data:

$$x_{DMD}(t) = \sum_{k=1}^{K} b_k\phi_k e^{\mu_k t} = \Phi diag(e^{\mu_k t})b \tag{10}$$

where K is the rank of X_1^{M-1}, b_k is the initial value of each mode, and Φ have eigenvectors of A as its column. At our initial condition $x_1 = \Phi b$ so we can easily get b by multiplyingh the pseudoinverse with x_1.

3 Algorithm Implementation and Development

Given the video, we first divide it into frames at each time. After that we can pick desired modes and use DMD to construct our background approximation. And instead of simply subtracting the background from the original data, we would want to fix negative values in the resulting foreground. Finally we can use the residual to have a edited background and foreground data. To implement this algorithm, we need following

Algorithm 1: Locating and extracting position of mass

Load data and convert video into 4-D matrix
Reshape the data matrix into 2-D matrix where each column represents the spatial data at the specific time point
Create X_1^{M-1} and X_2^M
Apply svd on X1 and pick first 2 modes
Use svd of X1 to find $\tilde{S}$ and construct background low rank approximation
Subtract the background from the original data to get the foreground
Find residual by fixing all negative values in the foreground and add to background

developments:

1. There are duplicates in frames with small time difference. To fix duplicates, I increase δt to sample parts of the frames with equal time interval and avoid failure in background separation due to duplicates.
2. The resulting foreground matrix has negative value, so it is important to find the residual and add those negative value back to the background.
3. To get our continuous time eigenvalue near 0, we pick first modes of our svd.

4 Computational Results

4.1 Monte Carlo

From Figure1 we can see two screen shots of original Monte Carlo race car video showing several cars passing by. There is almost no image of race cars appearing in the screen shots of separated background, so this is pretty successful. However, there is still some background image in the foreground video, but the race cars are shown fine and the separated video is overall pretty good.

4.2 Ski Drop

In the screenshots of original video of the ski drop, the player is noted in red circle showing it going from the top to the bottom. And we can see there is almost no image of this player in the screen shots of separated background, which indicates that this background separation is pretty successful. Besides, the player as well as its motion trace on the mountain are shown clearly and noted in red circle in the screenshots of separated foreground video. there is still some background but overall looks good.

Figure 1: Two screenshots of original Monte Carlo video

Figure 2: Two screenshots of separated background of Monte Carlo video

Figure 3: Two screenshots of separated foreground of Monte Carlo video

Figure 4: Two screenshots of original video of ski drop

Figure 5: Two screenshots of separated background video of ski drop

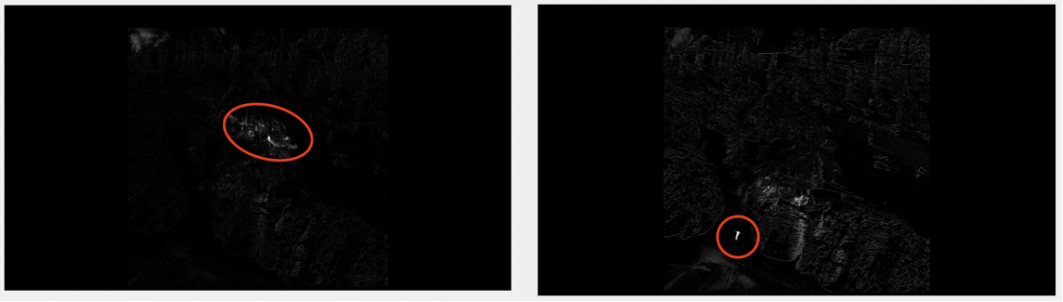

Figure 6: Two screenshots of separated foreground video of ski drop

5 Summary and Conclusions

Overall, we successfully use DMD to separate the background and foreground of a video with low rank approximation of desired modes. Changing the modes we use may provide better or worse results and can be compared for later studies.

Appendix A MATLAB Functions

MATLAB functions used in the implementations of Algorithms.

- `M = rgb2gray(A))` returns a 2d matrix `M` with value in gray scale of the rgb image stored in `A`.
- `M = reshape(A,[m,n])` returns a reshaped matrix `M` of size [m,n] with original entries stored in `A`.
- `[u,s,v] = svd(X, 'econ')` returns matrices U,S, and V using a simplified version of singular value decomposition.

- `[eV, D] = eig(A)` returns a matrix `eV` of eigenvectors of `A` and an diagonal matrix `D` storing eigenvalues of `A` along its diagonal.

Appendix B MATLAB Code

```
clear all; close all; clc

%% load data
v = VideoReader('monte_carlo_low.mp4');
frames = read(v);
[m,n,rgb,num_frames] = size(frames);
t = 1:10:num_frames;
dt = 10;
X = zeros(m*n,length(t));
for i = 1:length(t)
    single_frame = frames(:,:,:,(i-1)*dt+1);
    F = rgb2gray(single_frame);
    F = im2double(F);
    X(:,i) = reshape(F,[m*n,1]);
    imshow(F)
end
%% Create DMD matrices
X1 = X(:,1:end-1);
X2 = X(:,2:end);
%% SVD of X1 and Computation of ~S
[U, Sigma, V] = svd(X1,'econ');
U = U(:,1:2);
Sigma = Sigma(1:2,1:2);
V = V(:,1:2);
S = U'*X2*V*diag(1./diag(Sigma));
[eV, D] = eig(S); % compute eigenvalues + eigenvectors
mu = diag(D); % extract eigenvalues
omega = log(mu)/dt;
Phi = U*eV;
%% Create DMD Solution
y0 = Phi\X1(:,1); % pseudoinverse to get initial conditions
ubg_modes = zeros(length(y0),length(t));
for i = 1:length(t)
   ubg_modes(:,i) = y0.*exp(omega*t(i));
end
ubg_dmd = Phi*ubg_modes;
ufg_dmd = abs(X-ubg_dmd);
res = ufg_dmd.* (ufg_dmd < 0);
ubg_dmd = abs(ubg_dmd)+res;
ufg_dmd = ufg_dmd-res;
%%
for i = 1:length(t)
    frame_img = reshape(ubg_dmd(:,i),[m,n]);
    frame_img = mat2gray(frame_img);
    imshow(frame_img)
end
%%

for i = 1:length(t)
    frame_img = reshape(ufg_dmd(:,i),[m,n]);
    frame_img = mat2gray(frame_img);
    imshow(frame_img)
end



% %% Plotting Eigenvalues (omega)
%
% % make axis lines
% line = -15:15;
%
```