Chingpo Lin
Amath482
HW5

Abstract

There are two video for me to explore with DMD which is Dynamic Mode Decomposition. The two videos consist of two parts: background video and foreground video. I will use the low rank form and sparse form of X to explore them.

Introduction and Overview

I was given two kinds of video one is a video of moving car, and another is skiing. For each of video, I will separate them into foreground and background video, and get DMD solution of each of them. Also, I will separate them into low rank and sparse format, which will make pixel intensities non-negative, and having real-value on them. As a result, this method will definitely have a good performance.

Theoretical Background

Dynamic Mode Decomposition, which is DMD in short, which making use of low-dimensionality in experimental data without the use of a given set of governing equations. This will help us predict the future data. By using exponent function on time domain, DMD can actually find the basis of spatial mode. First thing is to setup DMD, which we define N, M, and time t with:

N = number of spatial points saved per unit time snapshot

M = number of snapshots taken.

$t_{m+1} = t_m + \Delta t,\ m = 1,\ldots,M-1,\ \Delta t > 0.$ (1)

And then we setup the snapshot which is

$$X_j^k = [U(x, t_j)\ U(x, t_{j+1})\ \dots\ U(x, t_k)] \text{ (2)}$$

With j denotes column, and will through k pf full snapshot matrix X.

Then, the Koopman Operator gives us:

$$x_{j+1} = A\, x_j \text{ (3)}$$

With A is the linear operator that mapping time from j to j+1, and x is n-dimensional vector that collected at time j

Then we come up with DMD, and above equation help us to construct:

$$X_1^{M-1} = [x_1\ A\, x_1\ A^2 x_1\ \dots\ A^{M-2} x_1] \text{ (4)}$$

By adapting SVD transformation, we can have:

$$U^\star A U = U^\star X_2^M V S^{-1} \text{ (5)}$$

Then, by some transformation, we will have our final equation:

$$\mathrm{x_{DMD}(t)} = \sum_{k=1}^{K} \mathrm{b_k \psi_k e^{\omega_k t}} = \Psi \mathrm{diag(e^{\omega_k t})b}. \text{ (6)}$$

Where bk are initial amplitude and matrix ψ contains eigenvector $\psi_{\mathrm{k.}}$

Then, we divide this into the sum of background and foreground part to make it work as expected:

$$\text{(7)} \qquad \mathrm{X_{DMD}} = \underbrace{b_p \varphi_p e^{\omega_p t}}_{\text{Background Video}} + \underbrace{\sum_{j \neq p} b_j \varphi_j e^{\omega_j t}}_{\text{Foreground Video}}$$

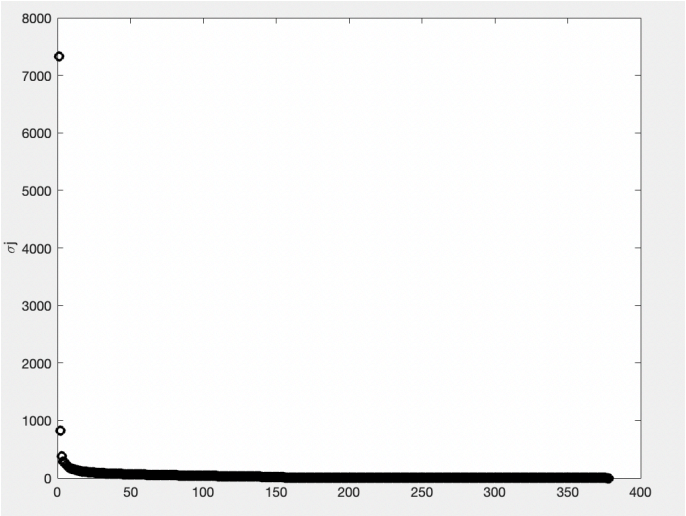
And we covert it into the sum of low rank and sparse:

$$\text{(8)} \qquad \mathbf{X} = \mathbf{X}_{\mathrm{DMD}}^{\mathrm{Low\text{-}Rank}} + \mathbf{X}_{\mathrm{DMD}}^{\mathrm{Sparse}},$$

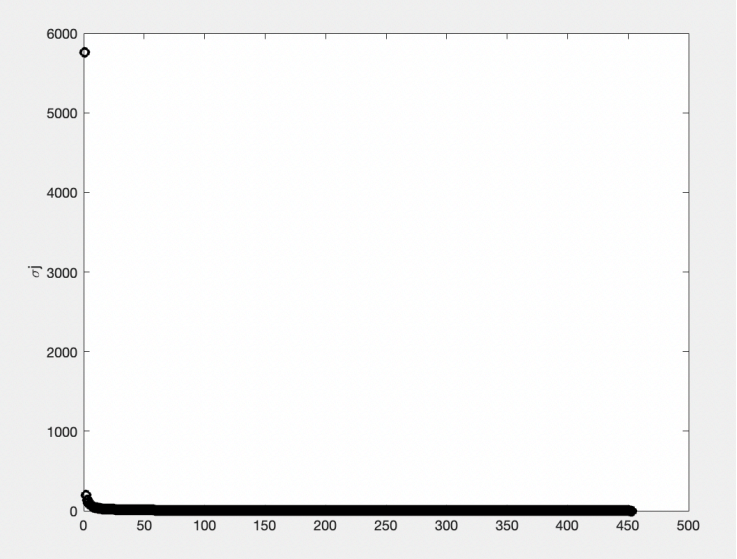
Algorithm Implementation and Development

First, I set up for time and all related condition, then, I remove the color in that video and turn it into double. Then, the dmd part starts, I first construct two submatrices and take the svd of first one. Then I create the S matrix by formula to find out its Eigenvalue and eigenvector. Then, I find out the pseudoinverse of Ψ and the coefficients bk, then I have all to compute the solution.

Computational Result

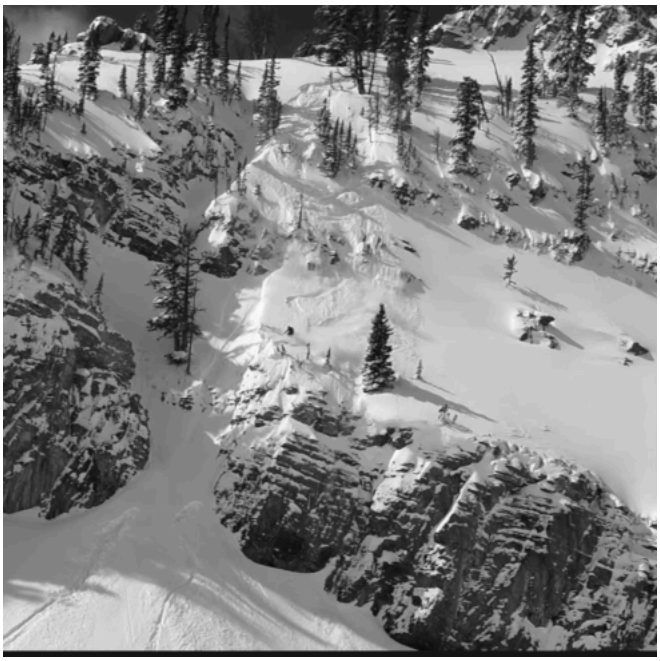


figure(1): energy plot of first video



figure(2): energy plot of first video

figure(3): snapshot of reconstructed video1



figure(4): snapshot of reconstructed video2

Summary and Conclusions

We see that the DMD indeed did a good job on dealing with video, it still has high quality after reconstruction into sparse and low rank combination.

Appendix A - function in code

video reader: read the video
im2double: convert image into double format
rgb2gray: remove the color of image
zeros: create a vector or matrix with all 0 in given dimension
svd: get the svd matrix decomposition

Appendix B - Matlab code

```
%% Clean workspace
clear all; close all; clc
%% import data1
v1 = VideoReader('monte_carlo_low.mp4');
%% import data2
v2 = VideoReader('ski_drop_low.mp4');
%% calculation1
dt1 = 1/v1.Framerate;
t1 = 0:dt1:v1.Duration;
vf1 = read(v1);
nf1= get(v1,'NumFrames');
f1 = im2double(vf1(:,:,:,1));
s1 = size(f1,1) * size(f1,2);
a1 = zeros(s1, nf1);

for j = 1:nf1
    f = vf1(:,:,:,j);
    f = rgb2gray(f);
    f = im2double(f);
    a1(:,j) = reshape(f, 540 * 960, []);
    % show the image
    % imshow(f); drawnow
end
%% calculation2
dt2 = 1/v2.Framerate;
t2 = 0:dt2:v2.Duration;
```

```
vf2 = read(v2);
nf2= get(v2,'NumFrames');
f2 = im2double(vf2(:,:,:,1));
s2 = size(f2,1) * size(f2,2);
a2 = zeros(s2, nf2);

for j = 1:nf2
    f = vf2(:,:,:,j);
    f = rgb2gray(f);
    f = im2double(f);
    a2(:,j) = reshape(f, 540 * 960, []);
    % show the image
    % imshow(f); drawnow
end

%% DMD starts here part1
ax11 = a1(:,1:end-1);
ax12 = a1(:,2:end);
[U1, Sigma1, V1] = svd(ax11,'econ');
plot(diag(Sigma1),'ko','Linewidth',2)
ylabel('\sigmaj')
S1 = U1'*ax12*V1*diag(1./diag(Sigma1));
[eV1, D1] = eig(S1); % compute eigenvalues + eigenvectors
mu1 = diag(D1); % extract eigenvalues
omega1 = log(mu1)/dt1;
Phi1 = U1*eV1;

y10 = Phi1 \ ax11(:,1); % pseudoinverse to get initial conditions
umodes1 = zeros(length(y10), length(t1));
for iter = 1:length(t1)
    umodes1(:,iter) = y10 .* exp(omega1*t1(iter));
end
u_dmd1 = Phi1 * umodes1;

%% DMD starts here part2
ax21 = a2(:,1:end-1);
ax22 = a2(:,2:end);
[U2, Sigma2, V2] = svd(ax21,'econ');
plot(diag(Sigma2),'ko','Linewidth',2)
ylabel('\sigmaj')
S2 = U2'*ax22*V2*diag(1./diag(Sigma2));
[eV2, D2] = eig(S2); % compute eigenvalues + eigenvectors
mu2 = diag(D2); % extract eigenvalues
omega2 = log(mu2)/dt2;
Phi2 = U2*eV2;

y20 = Phi2 \ ax21(:,1); % pseudoinverse to get initial conditions
umodes2 = zeros(length(y20), length(t2));
for iter = 1:length(t2)
    umodes2(:,iter) = y20 .* exp(omega2*t2(iter));
end
u_dmd2 = Phi2 * umodes2;

%% DMD calculation result1
```

```
xs1 = ax11 - abs(u_dmd1(:,size(u_dmd1, 2) - 1));
neg1 = xs1 < 0;
R1 = xs1 .* neg1;
un1 = R1 + abs(u_dmd1(:,size(u_dmd1, 2) - 1));
xsn1 = xs1 - R1;
rec1 = xsn1 + un1;
sf11 = size(f1,1);
sf12 = size(f1,2);
show1 = reshape(u_dmd1, [sf11, sf12, length(t1)]);

%% DMD calculation result2
xs2 = ax21 - abs(u_dmd2(:,size(u_dmd2, 2) - 1));
neg2 = xs2 < 0;
R2 = xs2 .* neg2;
un2 = R2 + abs(u_dmd2(:,size(u_dmd2, 2) - 1));
xsn2 = xs2 - R2;
rec2 = xsn2 + un2;
sf21 = size(f2,1);
sf22 = size(f2,2);
show2 = reshape(u_dmd2, [sf21, sf22, length(t2)]);

%% show for 1
for i = 1:nf1
   imshow(im2uint8(show1(:,:,i)))
end
title("Reconstruction sum of X_low_rank + X_Sparse");

%% result
L=40;n=379;
x2 = linspace (0,L,n+1);
x = x2(1:n);
subplot(2,1,1), waterfall(x,t1,abs(u_dmd1), colormap([0 0 0]))
xlabel('x')
ylabel('t')
zlabel(' | u | ')
title('DMD Solution')
set(gca,'FontSize',16)

%% show for 2
for i = 1:nf2
   imshow(im2uint8(show2(:,:,i)))
end
title("Reconstruction sum of X_low_rank + X_Sparse");
```