AMATH482 Homework 5

Vandy Zhang
March 17, 2021

Abstract

In this assignment, we are given two video clips, and we are asked to separate its foreground video and background video using the Dynamic Mode Decomposition method (DMD).

1 Introduction and Overview

For this homework, I was asked to apply the DMD method and separate two video clips into its foreground video and background video. I should first import the two videos and extract the matrix data from them. To get the multidimensional matrix after DMD, I should split the original matrix data to a low-rank matrix and a sparse matrix, then transform them by subtracting or adding the residual values of the sparse matrix, and then add them back. For the first step while getting the low-rank matrices, the singular value decomposition should also be applied.

2 Theoretical Background

2.1 The Singular Value Decomposition

While multiplying a vector by a matrix, we have $\mathbf{Av} = \boldsymbol{\sigma}\mathbf{u}$, in which $\boldsymbol{\sigma}$ represents a stretching vector and $\mathbf{u}$ represents a rotation vector. In other words, when we trying to change a vector, we change its direction by applying a rotation vector to it and change its length by applying a stretching vector to it. In matrix form, we have $\mathbf{AV} = \mathbf{U\Sigma}$, where $\mathbf{V}$ is a unitary matrix, $\mathbf{U}$ represents a rotation matrix, and $\boldsymbol{\Sigma}$ represents a stretching matrix with $\boldsymbol{\sigma}$ on its diagonals representing singular values. For any unitary matrices, we have $\mathbf{V^{-1}} = \mathbf{V^*}$, where $\mathbf{V^*}$ means the transpose of V. Therefore, we can isolate A to get:

$$\mathbf{A} = \mathbf{U\Sigma V^*},$$

thus, giving us a decomposition of the matrix A through unitary matrices $\mathbf{U}$ and $\mathbf{V}$ and a diagonal matrix $\boldsymbol{\Sigma}$. The principal components are the squares of the singular values.

In MATLAB, there is a built-in svd(X, 'econ') function that we can use to apply the reduced singular value decomposition, which is a more efficient way that has a lower order.

2.2 The Koopman Operator

The Koopman Operator is a linear, time-independent matrix such that

$$x_{j+1} = Ax_j$$

In which j represents the time information, and A represents the linear operator that maps x_j to x_{j+1}. [1]

2.3 The Dynamic Mode Decomposition

Considering the Koopman Operator, we can have our data sampled like this:

$$x_1^{M-1} = [x_1 \;\; Ax_1 \;\; A^2x_1 \ldots \;\; A^{M-2}x_1]$$

The DMD algorithm is to create a low-dimension approximation of the linear mapping for data with low dimensionality. Here are the steps of DMD: [1]

1. Create a sample data X with a size of N * M, which should be linearly spaced in time by Δt.

2. Construct two submatrices x_1^{M-1} and x_2^M from X.

3. Apply the SVD to matrix x_1^{M-1}.

4. Calculate the low-rank dynamics S_tilda

$$\tilde{S} = U * x_2^M V \Sigma^{-1}$$

And then calculate the eigenvalues and eigenvectors of S_tilda, which are DMD modes.

5. Use x1 and the pseudoinverse of ψ to find the initial conditions

$$\psi_k = Uy_k$$

6. Compute the low-rank DMD solution

$$X_{dmd}(t) = \sum_{k=1}^{K} b_k \psi_k e^{w_k t} = \psi diag(e^{w_k t}) b.$$

3 Algorithm Implementation and Development

To load the video clips, I used VideoReader and got the number of frames and the duration of the video. By dividing the duration by the number of frames, I got the value of Δt. Then I used read to read all video frames. Since the size of both videos are too big, which costs too much computer storage, I used imresize to decrease the size of the video to the one tenth of its original. With the resized video, I iterated through each of its frames, turned each array from rgb to gray and from unit 8 type to double by applying rgb2gray and im2double, and finally reshaped the data to have a size of [ ] * numFrames.

Having my data sampled, I followed the procedures of the DMD method. Firstly, I got two submatrices x_1^{M-1} and x_2^M from my sampled data. Secondly, I applied the SVD using svd and calculated the rank of the data by setting the threshold of energy captured to be 90%. Thirdly, I calculated S_tilda using the rank and the U, S, V given by svd. Fourthly, I applied the eig method to get the eigenvalues and eigenvectors of my S_tilda. Having eigenvalues on hand, I turned them to the log scale so that they could be continuous. Fifthly, I calculated my ψ using the eigenvectors and set up the initial conditions using ψ and x1. Finally, I calculated the exponential DMD modes and multiplied them by ψ to get my Xdmd.

After getting the low-rank DMD matrix, I should separate the foreground video and the background video as well as reconstruct the video by combining the transformed background video and the foreground video. I first got the sparse matrix by subtracting the absolute value of the low-rank matrix from x_1^{M-1}. Secondly, I found the residual negative values of the sparse matrix and initialized as R. Then I added the matrix R back to the absolute value of the low-rank matrix and subtract R from the sparse matrix to get my sparse matrix and low-rank matrix transformed. After that, the reconstructed matrix could be retrieved as the combination of the transformed low-rank matrix and the sparse matrix.

By reshaping all 6 matrices back to three dimensions, I could use imshow to visualize the performance of each of the 6 datasets.

4 Computational Results

4.1 The Ski-drop Video

After reshaping the original data to be one-tenth large, the data had a size of 23667 * 455. From the singular value decomposition result, the number of singular values captured over 90% of energy was 19, so the rank I used was 19.

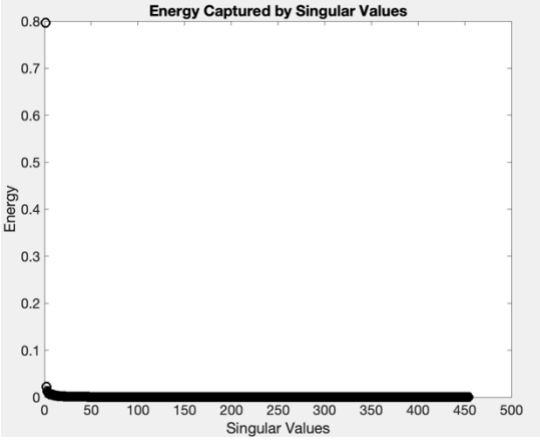

Figure 1: the plot of the distribution of the energy captured by the singular values of the first video Ski-drop

As seen in the Figure 2, compared to the original video, the reconstructed video performed pretty well. The background video before adding the residual negative values of the sparse matrix also performed well because the person skiing was invisible. However, compared to the upper middle video, the bottom middle video, which was the background video after adding the residual negative values of the sparse matrix performed worse because the person skiing was still visible. Regarding the foreground video, both performed very bad because the person skiing per se was very small and hard to see in the original video, so it was too hard for the foreground video to capture the feature. Therefore, the two foreground videos were almost black.

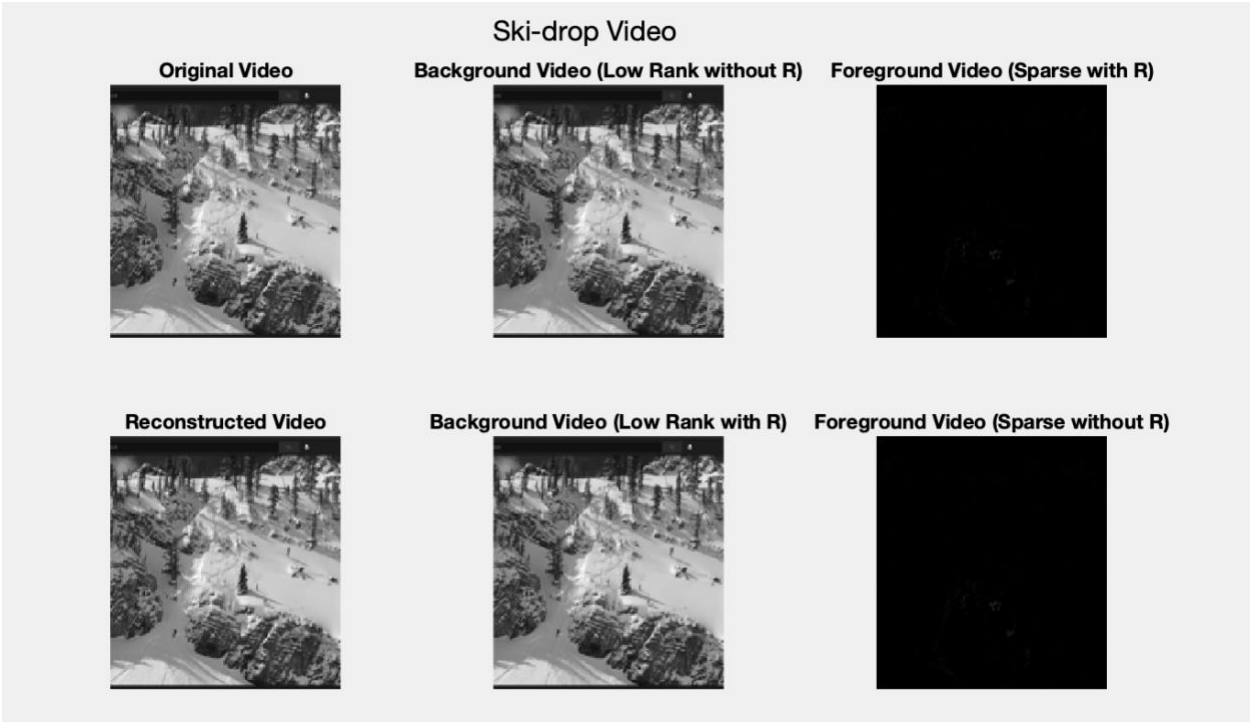

Figure 2: the plot of the 300th frame of the 6 videos extracted from the Ski-drop video. (Upper left) The original video of Ski-drop (Upper middle) The background video of Ski-drop before adding the residual negative values of the sparse matrix (Upper right) The foreground

video of Ski-drop before subtracting the residual negative values of the sparse matrix (Bottom left) The reconstructed video of Ski-drop (Bottom middle) The background video of Ski-drop after adding the residual negative values of the sparse matrix (Bottom right) The foreground video of Ski-drop after subtracting the residual negative values of the sparse matrix

4.1 The Monte-carlo Video

After reshaping the original data to be one-tenth large, the data had a size of 21056 * 380. From the singular value decomposition result, the number of singular values captured over 90% of energy was 92, so the rank I used was 92.

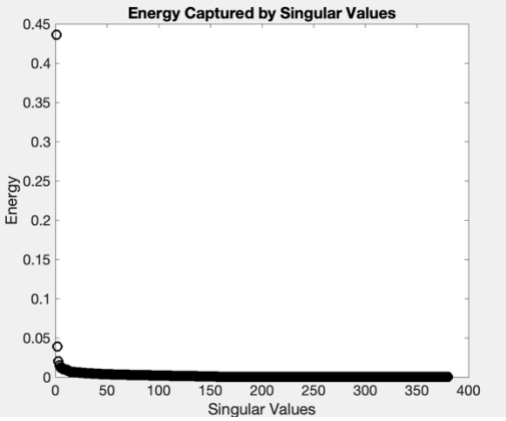

Figure 3: the plot of the distribution of the energy captured by the singular values of the second video Monte-carlo.

As seen in the Figure 4, compared to the original video, the reconstructed video performed pretty well. The background video before adding the residual negative values of the sparse matrix also performed well because the moving car was almost invisible. However, compared to the upper middle video, the bottom middle video, which was the background video after adding the residual negative values of the sparse matrix performed worse because there was still a shadow of the moving car. Regarding the foreground video, both performed quite well because there were outlines of the car and the bright spot on the car. Compared to the foreground videos of Ski-drop, these performed better because the moving item in the foreground per se was more obvious.

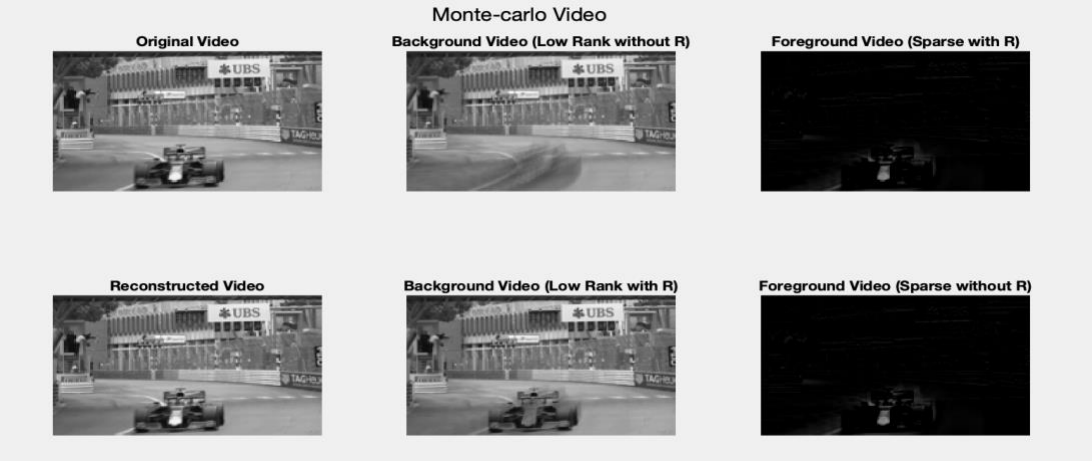

Figure 4: the plot of the 100th frame of the 6 videos extracted from the Monte-carlo video. (Upper left) The original video of Monte-carlo (Upper middle) The background video of Monte-carlo before adding the residual negative values of the sparse matrix (Upper right) The foreground video of Monte-carlo before subtracting the residual negative values of the sparse matrix (Bottom left) The reconstructed video of Monte-carlo (Bottom middle) The background video of Monte-carlo after adding the residual negative values of the sparse matrix (Bottom right) The foreground video of Monte-carlo after subtracting the residual negative values of the sparse matrix

5 Summary and Conclusions

To separate the two video clips to foreground videos and background videos, I applied the DMD method, which followed the principle of the Koopman Operator and used the SVD method as a pathway to find the number of the modes. All in all, the reconstructed data after DMD performed pretty well compared to the original data.

References

[1] University of Washington. AMATH482 Lecture 27 Notes. URL: https://canvas.uw.edu/courses/1433289/files/73620847?wrap=1

[2] Jose Nathan Kutz. Data-driven modeling & scientific computation: methods for complex systems & bigdata. Oxford University Press, 2013.

Appendix A. MATLAB Functions

reshape(A, sz)	reshape A to have a size of sz
VideoReader	create object to read video files
read	read all video frames
[U, S, V] = svd(A, 'econ')	perform the singular value decomposition and produces an economy-size decomposition of matrix A
[V, D] = eig(A)	return diagonal matrix D of eigenvalues and matrix V of eigenvectors
imresize (A, scale)	return an image that is the scale times of A
rgb2gray	convert RGB image to grayscale
im2double	convert the data type to be double
imshow	show the image

Appendix B. MATLAB Code

```
close all; clear; clc;
```

```
skiReader = VideoReader('ski_drop.mov');
numFrames = get(skiReader, 'NumFrames');
duration = get(skiReader, 'duration');
dt = duration/numFrames;

video = read(skiReader);
video = imresize(video, 0.1);
for k = 1:numFrames
    video_resized = im2double(rgb2gray(video(:,:,:,k)));
    skiVid(:, k) = reshape(video_resized, [], 1);
end

X1 = skiVid(:, 1:end - 1);
X2 = skiVid(:, 2:end);
[U, S, V] = svd(X1, 'econ');
sigma = diag(S);

k = 1;
while sum(sigma(1:k))/sum(sigma) < 0.9
    k = k + 1;
end

rank = k;

figure(1)
plot(sigma/sum(sigma), 'ko', 'Linewidth', 2, 'Markersize', 10);
title('Energy Captured by Singular Values');
xlabel('Singular Values'); ylabel('Energy');
set(gca,'Fontsize',16);

S_tilda = U(:, 1:rank)' * X2 * V(:, 1:rank)/S(1:rank, 1:rank);
[eV, D] = eig(S_tilda);
mu = diag(D);
omega = log(mu)/dt;
Phi = U(:, 1:rank) * eV;

t = 0:dt:duration;
y0 = Phi\X1(:, 1);

modes = zeros(length(y0),length(t)-2);
for iter = 1:length(t)-2
    modes(:,iter) = y0.*exp(omega*t(iter));
end
Xdmd = Phi*modes;

Xsparse = X1 - abs(Xdmd);
R = Xsparse.*(Xsparse < 0);
Xdmd_reconstructed = R + abs(Xdmd);
Xsparse_reconstructed = Xsparse - R;
Xreconstructed = Xsparse_reconstructed + Xdmd_reconstructed;

[row, col] = size(video_resized);
vid1 = reshape(X1, [row, col, numFrames-1]);
vid2 = reshape(Xdmd, [row, col, numFrames-1]);
vid3 = reshape(Xsparse, [row, col, numFrames-1]);
```

```
vid4 = reshape(Xreconstructed, [row, col, numFrames-1]);
vid5 = reshape(Xdmd_reconstructed, [row, col, numFrames-1]);
vid6 = reshape(Xsparse_reconstructed, [row, col, numFrames-1]);


figure(2)
subplot(2, 3, 1)
imshow(vid1(:,:,300))
title('Original Video');
subplot(2, 3, 2)
imshow(vid2(:,:,300))
title('Background Video (Low Rank without R)');
subplot(2, 3, 3)
imshow(vid3(:,:,300))
title('Foreground Video (Sparse with R)');
subplot(2, 3, 4)
imshow(vid4(:,:,300))
title('Reconstructed Video');
subplot(2, 3, 5)
imshow(vid5(:,:,300))
title('Background Video (Low Rank with R)');
subplot(2, 3, 6)
imshow(vid6(:,:,300))
title('Foreground Video (Sparse without R)');
sgtitle('Ski-drop Video');



close all; clear; clc;

monteReader = VideoReader('monte_carlo.mov');
numFrames = get(monteReader, 'NumFrames');
duration = get(monteReader, 'duration');
dt = duration/numFrames;

video = read(monteReader);
video = imresize(video, 0.1);
for k = 1:numFrames
    video_resized = im2double(rgb2gray(video(:,:,:,k)));
    monteVid(:, k) = reshape(video_resized, [], 1);
end

X1 = monteVid(:, 1:end - 1);
X2 = monteVid(:, 2:end);
[U, S, V] = svd(X1, 'econ');
sigma = diag(S);

k = 1;
while sum(sigma(1:k))/sum(sigma) < 0.9
    k = k + 1;
end

rank = k;

figure(1)
```

```
plot(sigma/sum(sigma), 'ko', 'Linewidth', 2, 'Markersize', 10);
title('Energy Captured by Singular Values');
xlabel('Singular Values'); ylabel('Energy');
set(gca,'Fontsize',16);

S_tilda = U(:, 1:rank)' * X2 * V(:, 1:rank)/S(1:rank, 1:rank);
[eV, D] = eig(S_tilda);
mu = diag(D);
omega = log(mu)/dt;
Phi = U(:, 1:rank) * eV;

t = 0:dt:duration;
y0 = Phi\X1(:, 1);

modes = zeros(length(y0),length(t)-2);
for iter = 1:length(t)-2
    modes(:,iter) = y0.*exp(omega*t(iter));
end
Xdmd = Phi*modes;

Xsparse = X1 - abs(Xdmd);
R = Xsparse.*(Xsparse < 0);
Xdmd_reconstructed = R + abs(Xdmd);
Xsparse_reconstructed = Xsparse - R;
Xreconstructed = Xsparse_reconstructed + Xdmd_reconstructed;

[row, col] = size(video_resized);
vid1 = reshape(X1, [row, col, numFrames-1]);
vid2 = reshape(Xdmd, [row, col, numFrames-1]);
vid3 = reshape(Xsparse, [row, col, numFrames-1]);
vid4 = reshape(Xreconstructed, [row, col, numFrames-1]);
vid5 = reshape(Xdmd_reconstructed, [row, col, numFrames-1]);
vid6 = reshape(Xsparse_reconstructed, [row, col, numFrames-1]);


figure(2)
subplot(2, 3, 1)
imshow(vid1(:,:,100))
title('Original Video');
subplot(2, 3, 2)
imshow(vid2(:,:,100))
title('Background Video (Low Rank without R)');
subplot(2, 3, 3)
imshow(vid3(:,:,100))
title('Foreground Video (Sparse with R)');
subplot(2, 3, 4)
imshow(vid4(:,:,100))
title('Reconstructed Video');
subplot(2, 3, 5)
imshow(vid5(:,:,100))
title('Background Video (Low Rank with R)');
subplot(2, 3, 6)
imshow(vid6(:,:,100))
title('Foreground Video (Sparse without R)');
sgtitle('Monte-carlo Video');
```