AMATH 482 Homework 5

Yiqing He

March 16th, 2021

Abstract

I used the Dynamic Mode Decomposition method on the video clips ski_drop.mov and monte_carlo.mov to separate the video stream to both the foreground video and a background.

Introduction and Overview

This assignment poses a problem when separating the DMD terms into approximate low-rank and sparse reconstructions: the DMD reconstruction is complex, but the real valued outputs are desired and knowing how to handle the complex elements can make a significant difference in the accuracy of the results. Doing all the maths, only negative values turned out. However, it does not make any sense of having negative pixel intensities. Finding a way to make pixel intensities to be higher or equal to zero is important. This could make the method work well.

Theoretical Background

In this project includes the implementation of Dynamic Mode Decomposition:

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

This method approximates the modes of the Koopman operator **A**, which is alinear, time-independent operator such that

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

where the j indicates the specific data collection time and **A** is the linear operator.

Algorithm Implementation and Development

I first calculated two submatrices for X and then compute the SVD of $\mathbf{X}_1^{M-1}$. Then I created a matrix $\tilde{\mathbf{S}} = \mathbf{U}^*\mathbf{X}_2^M\mathbf{V}\Sigma^{-1}$ to find the eigenvalues and eigenvectors. After that, I used the initial snapshot x1 to find the coefficients b_k. Therefore, I could compute the solution at any future time using the DMD modes along with their projection to the initial conditions and the time dynamics by using the eigenvalues of $\tilde{\mathbf{S}}$.

Computational Results

I was finding a hard time to calculate the S, V, D out, so there are no results available. I put the reasonable plot codes in the appendix.

Summary and Conclusions

Same as above.

Appendix A

v = VideoReader(filename) creates object v to read video data from the file named filename.
real(z) returns the real part of z. If z is a matrix, real acts elementwise on z.
Y = exp(X) returns the exponential *e*x for each element in array X. For complex elements z = x + iy, it returns the complex exponential

Appendix B

```
clear all;clc

mtcl = VideoReader('monte_carlo.mov')
skdp = VideoReader('ski_drop.mov')
numFrames = 0
data = cell([],1)
col = zeros(540*960, 379);

while hasFrame(mtcl)
    numFrames = numFrames + 1;
```

```
    mtcl2 = readFrame(mtcl);
    data(numFrames) = mtcl2(:,:,3);
    col(:,numFrames) =  reshape(data(numFrames),540*960,3);
end
x = usol'
X1 =X(:,1:end-1);
X2 = X(:,2:end);
[U, S, V] = svd(data, 'econ')
S = U'*X2*V*diag(1./diag(Sigma));
mu = diag(D);
omega = log(mu)/dt;
Phi = U*eV;
y0 = Phi\X1(:,1);
umodes = zeros(length(y0),length(t));
for iter = 1:length(t)
    umodes(:,iter) = y0.*exp(omega*t(iter));
end
udmd = Phi*umodes;
subplot(2, 1, 1)
plot(diag(S), 'ko', 'Linewidth', 2)
ylabel('\sigma_j')
set(gca, 'Fontsize', 16, 'Xlim', [0.9 delays+0.1])
subplot(2,1,2)
plot(t(1:end-delays+1), V(:,1), 'r', 'Linewidth', 2)
hold on
plot(t(1:end-delays+1),V(:,2),'b--', 'Linewidth', 2)
xlabel('t')
ylabel('v_j(t)')
set(gca,'Fontsize',16)

X1 = V(1:end-1,1:2)';
X2 = V(2:end-1,1:2)';

[U2, S2, V2] = svd(X1, 'econ');
Stilde = U2'*X*V2*diag(1./diag(S2));
[eV, D] = eig(D);
omega = log(mu)/dt;
Phi = U2*eV;
y0 = Phi\X1(:,1);
u_modes = zeros(length(y0), length(t));
for iter = 1:length(t)
    u_modes(:,iter) = y0.*exp(omega*t(iter));
end
u_dmd = real(Phi*u_modes);



while hasFrame(skdp)
    numFrames = numFrames + 1;
    skdp2 = readFrame(skdp);
    data(numFrames) = skdp2(:,:,3);
    col(:,numFrames) =  reshape(data(numFrames),540*960,3);
end
[U, S, V] = svd(data, "econ")
subplot(2, 1, 1)
plot(diag(S), 'ko', 'Linewidth', 2)
```

```
ylabel('\sigma_j')
set(gca, 'Fontsize', 16, 'Xlim', [0.9 delays+0.1])
subplot(2,1,2)
plot(t(1:end-delays+1), V(:,1), 'r', 'Linewidth', 2)
hold on
plot(t(1:end-delays+1),V(:,2),'b--', 'Linewidth', 2)
xlabel('t')
ylabel('v_j(t)')
set(gca,'Fontsize',16)

X1 = V(1:end-1,1:2)';
X2 = V(2:end-1,1:2)';

[U2, S2, V2] = svd(X1, 'econ');
Stilde = U2'*X*V2*diag(1./diag(S2));
[eV, D] = eig(D);
omega = log(mu)/dt;
Phi = U2*eV;
y0 = Phi\X1(:,1);
u_modes = zeros(length(y0), length(t));
for iter = 1:length(t)
    u_modes(:,iter) = y0.*exp(omega*t(iter));
end
u_dmd = real(Phi*u_modes);
```