AMATH482 HW04

Joshua Borgman

University of Washington

Josh Borgman

Thursday, February 26, 2015

Contents

1. Introduction and Overview

Principal Component Analysis is a technique that deconstructs a signal into representative modes that can be summed to recreate the original signal. The technique can be accomplished with a wide variety of basis functions. Fourier analysis is one such variant that is very common, decomposing a signal into sinusoidal components that capture the frequency content of the signal to a desired limit. A common technique in linear algebra for decomposing a particular signal, especially those represented in a matrix form, is eigenvalue decomposition. In this deconstruction the matrix is broken down into a collection of associated eigenvalues and eigenvectors.

In response to the somewhat limited techniques previously mentioned, a method designed to specifically give the best modal representation of a signal by any number of desired modes, namely single value decomposition (SVD), is used with great effect. Single value decomposition is a factorization of a matrix into a number of constitutive components, and represents a transformation that stretches/ compresses and rotates a given set of vectors. The SVD by construction is guaranteed to exist for any matrix making it of great utility in many applications.

Post decomposition a signal can be understood as a collection of modes which capture the majority of energy or variation in the system. It is in this way that we can analyze the principle modal components of a system and use the reduced representation for computational ease. The SVD in particular yields a set of proper orthogonal modes (POD) which aim to best capture the energy of a system in the fewest number of modes for any given reconstruction.

2. Theoretical Background

Basics of Singular Value Decomposition

The reduced single value decomposition represents a transformation $\boldsymbol{A}$ between two orthonormal basis $\boldsymbol{v}_j$ and $\boldsymbol{u}_j$, such that

$$\boldsymbol{A}\boldsymbol{v}_j = \sigma_j \boldsymbol{u}_j \quad 1 \leq j \leq n.$$

Or in compact matrix notation $\boldsymbol{AV} = \widehat{\boldsymbol{U}}\widehat{\boldsymbol{\Sigma}}$ or rearranging this to yield $\boldsymbol{A} = \widehat{\boldsymbol{U}}\widehat{\boldsymbol{\Sigma}}\boldsymbol{V}^*$**.**

Traditionally the reduced single value decomposition is augmented to construct a matrix $\boldsymbol{U}$ from $\widehat{\boldsymbol{U}}$, by adding and additional *m-n* columns that are orthonormal to the already existing set in $\widehat{\boldsymbol{U}}$, so matrix **U** then becomes an *m x m* unitary matrix. This is followed by adding an additional *m – n* silent rows. This leads to $\boldsymbol{\Sigma}$ having *r* positive diagonal entries, with the remaining *n – r* diagonals equal to zero. The full SVD decomposition then takes the form

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$$

And the following three matrices

$$\boldsymbol{U} \in \mathbb{C}^{m\times m} \; is \; unitary$$

$$\boldsymbol{V} \in \mathbb{C}^{n\times n} \; is \; unitary$$

$$\boldsymbol{\Sigma} \in \mathbb{R}^{m\times n} \; is \; diagonal$$

Useful theorems that concern the single value decomposition include the following

1. Every matrix $\boldsymbol{A} \in \mathbb{C}^{mxm}$ has a singular value decomposition for the form $\boldsymbol{A} = \boldsymbol{U\Sigma V}^*$, for which the singular values (σ_j) are uniquely determined and if A is a square and the singular values distinct, the singular vectors $\{u_j\} and \{v_j\}$ are uniquely determined up to a complex sign.
2. *If the rank of **A** is r, then there are r nonzero singular values.*
3. *range(**A**) =* $\langle \boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_r \rangle$ *and null(**A**) =* $\langle \boldsymbol{v}_{r+1}, \boldsymbol{v}_{r+2}, \dots, \boldsymbol{v}_n \rangle$
4. $\|\boldsymbol{A}\|_2 = \sigma_1$ and $\|\boldsymbol{A}\|_F = \sqrt{\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_r^2}$ where these are the L2 and Frobenius norms respectively. The Frobenius norm contains the total matrix energy, while the 2-norm definition contains the energy of the largest singular value.
5. *The nonzero singular values of **A** are the square roots of the nonzero eigenvalues of **A*A** (or **AA***).*
6. *If **A=A*** (self adjoint), then the singular values of **A** are the absolute values of the eigenvalues of **A**.*
7. *For* $\boldsymbol{A} \in \mathbb{C}^{mxm}$, *the determinant is given by* $|\det(A)| = \prod_{j=1}^{m} \sigma_j$. *This result stems from the fact that the matrices **U** and **V** are unitary matrices.*

Based on these theorems, we can understand the SVD as a sort of least-square fitting algorithm, allowing us to project the matrix onto low dimensional representations in a formal way.

More formally, we can represent the sum of r rank-one matrices as

$$\mathrm{A} = \Sigma_{j=1}^{r} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^*,$$

And partial sum which considers N, such that $0 \leq N \leq r$, can be represented as

$$\mathrm{A_N} = \Sigma_{j=1}^{N} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^*$$

Thusly, if $N = \min\{m, n\}$, we can define $\sigma_{N+1} = 0$, and

$$\|A - A_N\|_2 = \sigma_{N+1}$$

Finally it is worth emphasizing that the SVD produces characteristic features (principal components) that are determined by the covariance matrix of the data.

Eigenvalues, Eigenvectors and Diagonalization

The simpler alternative to SVD most related to linear algebra is the eigenvalue decomposition. In particular eigenvalues and eigenvectors have many applications in understanding differential equations. The equation

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{A}\boldsymbol{y}$$

can be assumed to have a solution of the form $y = \exp(\lambda t)$ which leads the eigen value problem:

$$\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{x}$$

Which we will solve by reforming the equation as follows.

$$\boldsymbol{A}\boldsymbol{x} - \lambda \boldsymbol{I}\boldsymbol{x} = (\boldsymbol{A} - \lambda \boldsymbol{I})\boldsymbol{x} = 0$$

We can thus find the eigenvalues (λ_j) and the associated eiegenvectors $\boldsymbol{x}$ which satisfy the eigenvalue problem.

One major benefit of the eigenvalue decomposition is its ability to decompose a matrix (here **A**) into the following form

$$\boldsymbol{A = S\Lambda S^{-1}}$$

Where $\boldsymbol{S}$ the matrix here is the columns are the eigenvectors of **A,**

$$\boldsymbol{S = (x_1, x_2, \dots, x_n)}$$

And the matrix $\boldsymbol{\Lambda}$ is the diagonal matrix whose diagonals are the corresponding eigenvalues, namely:

$$\begin{pmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n \end{pmatrix}$$

This diagonalization leads the useful property that $\boldsymbol{A^M = S\Lambda^M S^{-1}}$

Principal Component Analysis (PCA)

Principal Component Analysis, in combination with techniques such as SVD are widely used in dimensionality reduction of data. PCA offers users a low-dimensional projection of the viewed from its most informative viewpoints. This is accomplished by utilizing the first few principal components of the data so that the dimensionality of the transformed data is greatly reduced.

Proper Orthogonal Modes /Principal Components

The proper orthogonal modes are the representative modes generated from a single value decomposition. These modes are based on the L2 fitting of the covariance matrix of the data set. This unique set of basis functions are all orthogonal and represent the principal components of the analyzed system.

3. Algorithm Implementation and Development

Extracting Position

The paint can position was derived by searching for the flashlight, represented by a very high RGB value. In particular the searching was done in the 'red' domain, searching for pixels with an intensity of 255 and averaging their pixel locations. You can see plots of the position below. To limit the search area and the effect of outlying noise, we visually estimated the position of the paint bucket and cropped the image around the area to cut out outside noise.

Filtering Data

The shaky nature of the camera data caused difficulty in extracting the position in most of the cases. In order to filter the data, we can by eliminating outside noise in movements. To this end, we cropped the image to a narrow window around the paint can (here we used a window width of 100 pixels).

Principal Component Analysis and Reconstruction

The principal component analysis was performed using the following generic steps:

1. Organize the data into a matrix $\boldsymbol{A} \in \mathbb{C}^{m\times n}$ where *m* is the number of measurement types and *n* is the number of measurements (or trials) taken from each probe.
2. Subtract off the mean for each measurement type or row $\boldsymbol{x}_j$.
3. Compute the SVD and singular values of the covariance matrix to determine the principal components.

After determining the proper orthogonal modes (POMs), we can view the components in their uniqness, as well as combine a limited number of POMs to recreate the signal.

Single Value Decomposition

The Single value decomposition is accomplished through the use of the svd command which decomposes the vector into the matrices previously discussed. Namely:

$$[U, S, V] = svd(A);$$

Yields the single value decomposition for A, with basis vectors given by U and V, and diagonal singular value in S. To assemble the matrix A for our operation we used the form in which the rows are given by each x,y combination and the columns are the subsequent time measurements. More specifically

$$\begin{pmatrix} \boldsymbol{x_1} \\ \boldsymbol{y_1} \\ \boldsymbol{x_2} \\ \boldsymbol{y_2} \\ \boldsymbol{x_3} \\ \boldsymbol{y_3} \end{pmatrix} \rightarrow t$$

Using this form we are able to compute the svd for a given camera set.

4. Computational Results

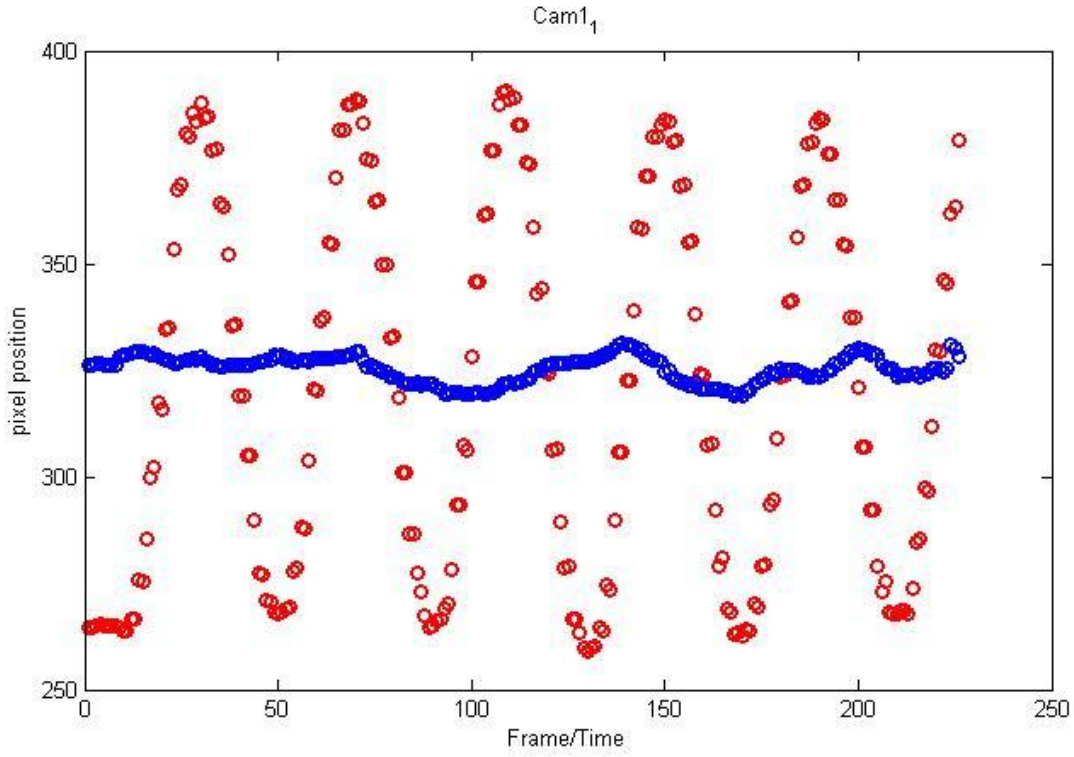

Figure 1: This is the position data extracted from camera 1_1; it shows very clear oscillatory motion in the up and down direction (red) and little movement in the horizontal direction (blue).

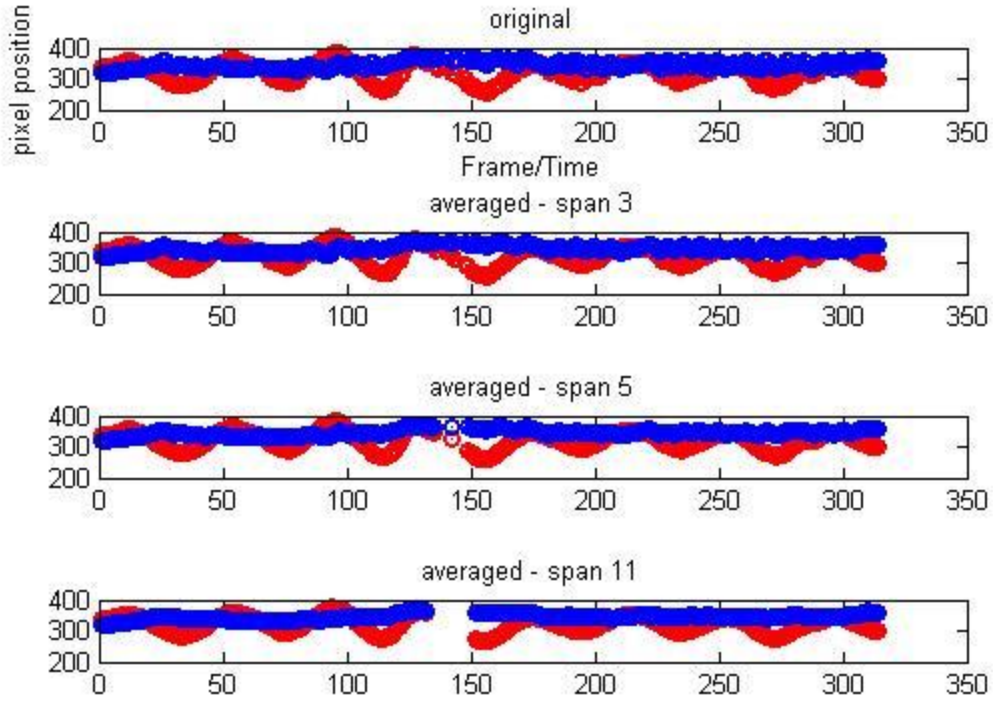

Figure 2: For poorly stabilized camera data such as that in camera 1_2, we can use a smoothing technique of applying a running average. We considered the effect of different spans of running average.

5. Summary and Conclusion

Principal component analysis provides a way to represent a large data set by a series of orthogonal modes. This can greatly simplify the data set for many beneficial applications. There is a danger in the proper application and usage of this powerful technique though, as it can be misused in inappropriate manners. Further this example highlighted the need for more computationally efficient mean especially when cleaning the noise. Such large data sets require an amount of memory that is often unavailable.

Appendix A:

Appendix B:

```
cd 'cam data';
obj=VideoReader('cam1_1.mat');
cd ..;


%% Load image and size info
clear all; close all; clc;
load('cam1_1.mat');
msize = size(vidFrames1_1);
vidFrames = vidFrames1_1;
numFrames = msize(4);

%% Loads Camera Two data -
clear all; close all; clc;
load('cam1_2.mat');
msize = size(vidFrames1_2);
vidFrames = vidFrames1_2;
vidTemp = double(vidFrames);
numFrames = msize(4);

%%
clear all; close all; clc;
load('cam1_3.mat');
msize = size(vidFrames1_3);
vidFrames = vidFrames1_3;
vidTemp = double(vidFrames);
numFrames = msize(4);
%%
red = double(vidFrames);
red(:,:,2:3,:) = 0;
center = msize(2)/2; window = 50;
red(:,[1:(center-window) (center+window):end],:,:) = 0;

%%
green = double(vidFrames);
center = msize(2)/2; window = 100;
green(:,:,[1 3],:) = 0;
green(:,[1:(center-window) (center+window):end],:,:) = 0;

green = uint8(green);
%%
blue = double(vidFrames);
center = msize(2)/2; window = 100;
blue(:,:,1:2,:) = 0;
blue(:,[1:(center-window) (center+window):end],:,:) = 0;

blue = uint8(blue);
%%
```

```
xpos = zeros(numFrames, 1);
ypos =zeros(numFrames, 1);
for j= 1:numFrames
    [x, y, z, t] = ind2sub([msize(1) msize(2)], find(red(:,:,1,j) >= 255));
    xpos(j)= mean(x);
    ypos(j)= mean(y);

end

%%
xsm = mySmooth(xpos);

%%
span = 11; ed = length(xpos); l = floor(span/2);
xposSm = zeros(length(xpos),1);
yposSm = zeros(length(ypos),1);
for jj =[1:l (ed-l):ed]
    xposSm(jj) = xpos(jj);
    yposSm(jj) = ypos(jj);
end


for j=1+l:(ed-l)
        xposSm(j)= mean(xpos((j-l):(j+l)));
        yposSm(j)= mean(ypos((j-l):(j+l)));
end

%%
subplot(4,1,4)
plot(1:numFrames, xposSm, 'ro-',1:numFrames, yposSm, 'bo-', 'Linewidth',
[2]), title('averaged - span 11');

%%
subplot(4,1,1)
plot(1:numFrames, xpos, 'ro-', 1:numFrames, ypos, 'bo-', 'Linewidth', [2])
title('original'); xlabel('Frame/Time'); ylabel('pixel position')
%% Used to build movie

for k = 1 : numFrames
    mov(k).cdata = red(:,:,:,k);
    mov(k).colormap = [];
end
%%

plays previously made movie matrix
for j=1:numFrames
    X=frame2im(mov(j));
    imshow(X); drawnow
end
%% creates "movie matrix"
for k = 1 : numFrames
    mov(k).cdata = vidFrames(:,:,:,k);
    mov(k).colormap = [];
end
```

```
%%plays previously made movie matrix
for j=1:numFrames
    X=frame2im(mov(j));
    imshow(X); drawnow
end

%% Creates avi image to watch outside matlab
vidObj = VideoWriter('test1_1RED.avi');
open(vidObj);
for j=1:numFrames
    X3=frame2im(mov(j));
    writeVideo(vidObj,X3);
end
close(vidObj);
```