Low Dimension Approximations using Principal Component Analysis for Analysis of a Spring-Mass System

Constance Wang

Abstract

In this problem, low dimension approximations for high dimension data is done using Principal Component Analysis. High dimension data is given in the form of videos taken of a spring-mass system in oscillatory motion. This motion is easily interpretable as one-dimensional, but there was oversampling done on this data. Capturing this motion is in multidimension since there was so much oversampling, but using lower dimension approximations from PCA, the original one-dimensional motion can be captured.

Section I – Introduction and Overview

Principal component analysis (PCA) is a tool to understand high dimensional data by extracting the most important information. It is used in a variety of problems, from fluid flow analysis to image processing, and is typically understood as a statistical tool. In this problem, the mechanics of a simple spring-mass system are recorded by different cameras. In these recordings, a mass (a paint can) is attached to the end of a spring, which oscillates in a predictable manor (given by laws of physics). To the viewer, this relationship obviously only exists in one dimension. However, we are given 12 recordings (four different test conditions) of this oscillation motion. The goal is to determine if this one-dimensional motion can be captured using PCA.

In more practical applications of PCA, relationships in the data may not be so obvious to the viewer. In these cases, large amounts of data (which may be potentially redundant) needs to be collected to ensure that the behavior of the system will be captured in its entirety. This problem reflects this situation in practical applications. A large amount of data, most of which is redundant, is captured and the system is oversampled. Using PCA, the most ideal, and simplest behavior can be extracted making it a powerful tool in performing accurate, low dimension approximations.

Section II – Theoretical Background

The underlying concept behind PCA is the SVD. The connection between the SVD and PCA is not immediately clear, so to understand the PCA, an understanding of SVD is necessary.

Singular Value Decomposition (SVD)

The purpose of the SVD is to literally decompose a matrix into different constituents. One understanding of a matrix is that it is a linear transformation, changing the direction and/or magnitude of a vector. This can also be understood as moving the vector from a space $\mathbb{R}^n$ to $\mathbb{R}^m$. In particular, there are n vectors ($\mathbf{v}$) living in $\mathbb{R}^n$, that when transformed will produce a particular set of orthogonal vectors $\mathbf{u}$ (there are m different $\mathbf{u}$ vectors) stretched by some factor (often called σ, and there are m different σ values). The SVD is given by Equation 1:

$$A = U\Sigma V^* \qquad \text{Equation 1}$$

where U is a matrix of the transformed vectors, $\mathbf{\Sigma}$ is a diagonal matrix where the stretching factors σ (often known as singular values) are the diagonal elements ordered from largest to smallest, and V^* is the transposed matrix of original vectors. These transformed vectors and their stretching factor have some special properties that will be described in more detail below. In addition, there is typically a

distinction between the different definitions of the SVD, namely, the reduced SVD and the full SVD. The equation above is the full SVD, which computes the SVD taking U to be (m x m) instead of (m x n). In this problem, the SVD is taken of simply the (n x n), so only the first n of the original vectors are taken.

Covariance

To understand how to use the SVD more practically, covariance must be used. This is a statistical concept that can help to remove redundant data. Covariance is the measure of how correlated two variables are, i.e. how statistically dependent/independent they are. This is given by the Equation 2:

$$\sigma_{ab}^2 = \frac{1}{n-1} ab^T \qquad \text{Equation 2}$$

In practice, covariance can be determined of all combinations of multiple variables using a covariance matrix, defined below in Equation 3:

$$C_x = \frac{1}{n-1} XX^T \qquad \text{Equation 3}$$

The diagonal terms of the covariance matrix are the variances of the individual variable, while the off-diagonal terms are the covariances between two variables. High individual variance means that the variable has values that are very spread, and vis versa. High covariance means that the two variables in question are statistically dependent (meaning that you can easily derive one from the other) and vis versa. This means that high covariance indicates redundancies in the data. To remove this redundancy, the mathematic concept of diagonalization needs to be used for the covariance matrix.

Diagonalization with the SVD

The power of SVD is captured in the concept of diagonalization. The result of doing the SVD diagonalizes the covariance matrix, which is exactly what is needed. Diagonalization of the covariance matrix means that the diagonals of the matrix will be ordered from largest to smallest (note the similarity with Σ in the SVD) and the off diagonals will be zero (meaning that all the variables are statistically independent, effectively removing redundancies).

Ordering the diagonals from largest to smallest indicates that the first diagonal element has the largest variance. The data with the largest variance are of key interest because these are the data with the most fluctuations. For this purpose of this problem (and most other problems), it becomes clear that the data with the most fluctuation is the "data of interest": it will be the direction the mass on the spring will oscillate in.

There are many ways to diagonalize, but in this problem, the SVD will complete diagonalization of the matrix. In this process, in addition to determining the "data of interest", a low dimension orthonormal basis will be determined. This is an ideal basis in which the covariance matrix is diagonalized (similar to the vectors of). The mathematic details will be spared, but the Matlab code is given (see Appendix B).

Section III – Algorithm Implementation and Development

The challenge for this problem is to determine how to collect the position data of the oscillating mass. Since the final result was to use PCA to find some meaningful data, the final data matrices needed to be in a proper format to take the SVD.

One of major components for this problem were to make sure the data was all the same length so that SVD could be completed. The videos were shortened manually by lining up the different frames to the correct starting and ending positions for the mass (see Appendix B: Manually determine length of video needed).

In addition, there were challenges posed by the different test conditions (labeled 1-4). For this problem, there were two steps taken to isolate the data and collect meaningful results: first to isolate the region in which the mass was oscillating by using a filter function, and then to find a way to determine the motion of the mass. Different videos called for different methods to be used. To get the data in the proper format, the im2double( ) command was used so that further operations, such as finding the average frame, could be performed. Then, either no modifications, removing the average frame, or manual input was used to track the path of the mass.

For no modifications and removing the average frame (automated approaches), the goal was to isolate a region in which the paint can was moving (roughly determining this when playing the video using implay( )) by creating a filter with a window that turns all pixels outside of the window to black. Each frame was modified in this way. This was done by using ginput( ) to manually determine the boundaries of the window. With removing the average frame, each frame was further modified to subtract the average frame (across the entire video) from each frame. This helped remove some noise generated. After determining the window boundary and/or removing the average frame, the maximum pixel value was determined. The index of this maximum was determined so that its location in the frame could be recorded. The maximum value was taken because often this value is the value of the flashlight on top of the mass. Often, the maximum pixel occurs in multiple locations (the flashlight is larger than one pixel), so the average position was taken. This workflow worked for the majority of the videos, but for the shaking cameras, the data was quite noisy.

For the shaking cameras, manual entry was used again with ginput( ). For every given number of frames, the program would record the input location (i.e. the user would click the location of the mass) and the previous input location, and do linear interpolation with linspace( ) to determine the location of the can in between.

For the PCA analysis, the data for each test was recorded in four separate matrices (since the data has different lengths). An SVD is taken of all these data using the svd( ) command and the 'econ' option to perform the reduced SVD. To determine the "data of interest", or the most important direction the mass is moving in, the variance of the data can be calculated by calculating the "energy" of the matrix (described further in Section IV).

Section IV – Experimental Results

For the determining the location of the mass using the automated approaches, Figure 1 shows in general how the paint can locations were generated.

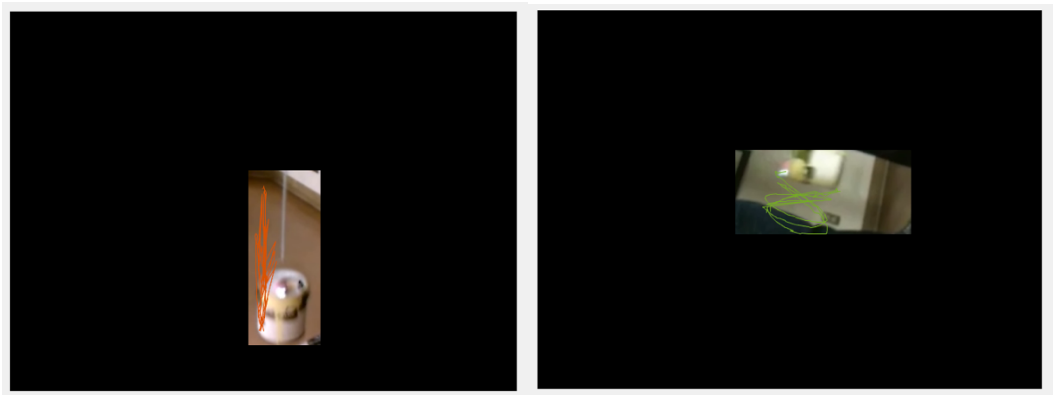

Figure 1: Capturing the paint can motion and visualizing the motion for Test 1_1 (left) and Test 3_3 (right)

Note that in Figure 1, the colored line in the picture indicates the total motion of the can, which is reasonable in Test 1 since there is not much noise in the data. On the other hand, for Test 3, it is quite obvious that the motion of the paint can is in multiple dimensions. However, after taking PCA on the data, these main motion of the paint can still be captured. This is shown in Figure 2.

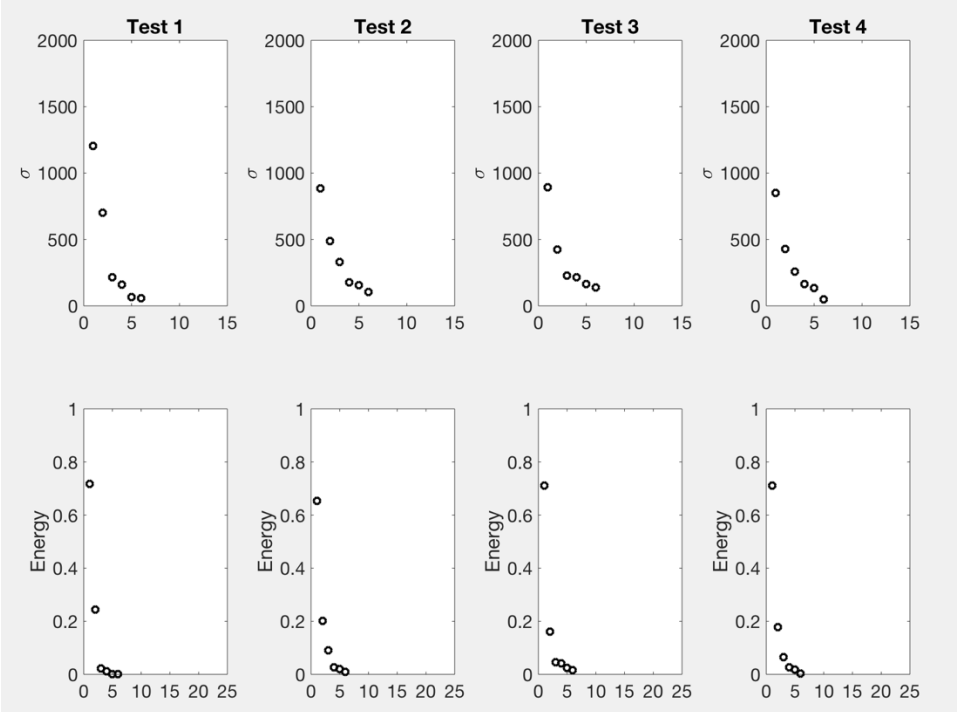

Figure 2: Determining the magnitude of the singular values (σ) in the first row of the data, and the "energy" of the matrix

The major pieces of information given in Figure 2 is that the singular values should determine which mass motion directions are the most important. In Test 1, the first singular value is greater in magnitude compared to the other tests, which makes sense because the data is definitely the cleanest in this test. This means that the motion of the mass can be mostly captured using the first singular value (indicating one dimensional motion).

The energy of the matrix is shown in the second row of Figure 2. These "energies" are a measure of how much that particular principal component contributes to the approximation of the system in question. The first principal component can approximate the system very well. The energy is measured as a fraction of the total original energy of the matrix. There is a comparison between the principal component (or the singular values) and how well this principal component can capture all the information of the original matrix. Therefore, in most of these tests, the captured energy is similar most of the principal components

Section V – Summary and Conclusions

Through this problem, it can be determined that PCA is a powerful analysis tool to determine the important details of a system. In this particular example, the different "tests" of the motion of the paint can were used to test the robustness of the PCA as a tool to remove any redundancies in the data, as well as any potential noise. For removing noise, this was done manually (since the original results from automated testing were poor) with the data collection techniques mentioned. However, for removing redundancies, the PCA is able to remove the redundant data given that within just the first principal component, almost all the information was able to be captured (about 80%). In addition, not all noise in the data was removed (for example, with the test 3 the swinging paint can). This shows that PCA can be a reliable tool in both noise and redundancy removal to determine the most important information from a given set of data.

Appendix A

- **implay( )** – this function was used to play the videos downloaded into Matlab, showing even the frame count of the video and which frame the video was on
    - function used to see the location of the paint can to determine filter window width but also to count frames
- **im2double( )** – this function is used to convert the uint8 matrices into double-precision values to perform calculations, such as finding the average frame in this problem
- **imshow( )** – this function is used to show an image knowing magnitudes of colors (in uint8 format) and its pixel location (size of matrix)
    - this function was often used to verify that the data collected was valid, since data of paint can location could be plotted on top
- **[x,y] = ginput( )** – this function takes a mouse click on an image and returns the location (pixel x and y value) of the click, which was used to determine the paint can location in manual input and also the window size
- **[U, S, V] = svd(X, 'econ')** – this function performs the SVD on a matrix, which is used to determine the principal components of the different tests
- **ind2sub( )** – the index of given by the max command is in linear coordinates, so this command changes the index to a 3D index location so that we can locate this value within each of the spatial and/or frequency dimensions