Homework 6: Background Subtraction in Video Streams

Karl Marrett (kdmarrett@gmail.com)

Abstract

The task in this report is to separate foreground and background objects in a video stream. To do this we will compare two strategies: the Dynamic Mode Decomposition and the robust Principal Component Cdoes for their performance in this task. The goals of the report are two separate the objects with these two methods and compare them qualitatively.

1 Introduction

We have thus far used several methods for taking high dimensional systems and projecting them into a lower perhaps more tractable form. The Dynamic Mode Decomposition takes a similar inspiration and returns the *dynamic* modes that comes from data that changes with respect to time.[?] Compared to our previous techniques, this basically provides a formal way to analyze linear structure in time as opposed to single snapshots in a data set. Given the problem of separating two objects with different linear dynamics in video frames, the motivation for this technique should be clear. This algorithm is defined more rigorously in the next section.

Finding the right basis and the methods to transform data into such a basis are central problems in machine learning. In the past reports, we have mostly used certain assumptions about our data to decide a reasonable basis to transform our data. For example, in the analysis of sound files we used the convenient decomposition into the Fourier domain to create a low dimensional picture of the dynamics of the data.[?] However, when we instead don't know anything about the underlying structure of mechanism of the data, we need a systematic approach. For example, given neural data of a spike train response of several neurons we may be able to explain structure in the data in a low dimensional Fourier space but the simplest model may in fact be described in an entirely different space. For problems of this nature, we want a data-driven approach for choosing the right basis to describe our data. Having a generic to high dimensional data of this sort allows equation-free modeling while also sidestepping our potentially misleading initial assumptions about the data.

These concerns illustrate some of the motivation for new methods for choosing a lower dimensional space for the data. One approach of choosing a basis involves the technique

known as Singular Value Decomposition, denoted as SVD henceforth in the report. The information that the SVD produces is quite similar to the eigen decomposition of the covariance matrix used in the previous report, in that the principal modes of the matrix are returned. Further details of how this technique might describe a low dimensional bases for our data is described in detail in the next section.

2 Theoretical Background

The DMD technique requires certain assumptions about our underlying system and the data being processed. The data must be collected at regularly spaced intervals of time, this is essentially builds the N dimensional matrix we define as $\mathbf{X}$ to compute future time points. Although the sampling must be linearly, the amount of sampling will have particular effects on the analysis which we will discuss more in the Section **??**. We build the data matrix X of N data points and M time points in the following way:

$$\mathbf{X} = [\mathbf{U}(\mathbf{x}, \mathbf{t_1}), \mathbf{U}(\mathbf{x}, \mathbf{t_1}), \mathbf{U}(\mathbf{x}, \mathbf{t_1}), \dots \mathbf{U}(\mathbf{x}, \mathbf{t_M})]$$

where $\mathbf{x}$ is the original data matrix of N channels. We must further define another matrix that is a subset of the columns of X as shown below:

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

This is just the subset of columns j through k of the original X matrix.

2.1 The Koopman Operator

The Koopman Operator relates the dynamics of one time slice of our original m channeled $\mathbf{X}$ matrix to a future time point. A few things to note about the Koopman operator relavant to the DMD are:

1. it is a linear and time-independent operator
2. it represents nonlinear, infinite dimensional dynamics without linearization
3. it can be viewed as the low dimensional modes

The dynamic modes of the decomposition are those of the Koopman operator.

2.2 SVD

The SVD has a particular role in the DMD algorithm.

The main means of compressing and characterizing the high dimensional song data is via the SVD technique. The theorem introducing the SVD essentially asserts that *any* matrix can be decomposed as follows:

$$A_m n = U_m m S_m n V_n n^T$$

The decomposition has deep connections to the eigen faces of A, namely in that the columns of U are orthonormal eigenvectors of AA^T and that the columns of V are also the orthonormal eigenvectors of A^TA. Another point to note is that the diagonal matrix S holds the square roots of the eigenvalues of U or V. These relations allow us to analytically solve for the u, s, or v matrices although in this report we will only be leveraging the Matlab command *svd*. Since U represents the new modes of A we can treat SV^T as the projection on to the modes. We take these projections as the means to classify the song data.

When implementing the SVD in the script we need to use the reduced SVD to the size of the data. The major advantage of the SVD is the ordered components of the weights of the S matrix, which gives a criterion for deciding which dimensions describe meaningful structure in the data, and those which are redundant.

3 Algorithm Implementation and Development

As alluded in the theoretical background section, the brunt of the mathematical sophistication of the identification techniques was handled by the *svd*, *classify*, and also the *spectogram* functions in Matlab. This means that the brunt of the complexity of the code was dealing with cleaning and ordering the data accordingly. For each of the three sections of the report the music files were put into a characteristic folder. The function *getSpec* written for this project conveniently batch processed every music file in the directory given. Processing the sound files in a separate function also allowed clearance of variables so that memory loads did not occur. For every sound file in the directory, a random sampling of ten three second clips was taken. From these files, the function *spectogram* was used to compute the changes in frequency over time. My gabor function developed from earlier reports was originally used but the timing of the matlab embedded *spectogram* was better, so it was used in the final analysis. Using spectogram data accentuates certain characteristics of the music. For example, while the raw trace would capture timing and patterns in beat, it would not capture spectral information such as the pitch range of the band or genres as strongly, which is likely key for a discrimination task of this sort. For these reasons, a spectogram was an ideal preprocessing step to give the SVD decent traction on the songs provided. During this process the song was downsampled to save space and the data was reshaped so that each clip became essentially one column to deal with. Treating a clip as a single vector made computations further in the script much easier. For each of the three categories of music, a variable number of usable vectors were produced but I needed a constant number from each group. To combat this I took the minimum number of samples from the three groups for the rest of the computations. The data returned by *getSpec* was a matrix with the rows defined by the spectogram of a song sample and 1 column for each clip.

Each of the sets of columns for each part was concatenated then run using the reduced SVD which returned the modes u, the singular values s and the data's projections onto the modes v. For the classification remainder of the script, the corresponding elements of the v vectors were split into three respective categories. Of these categories, the trials were split in training and test trials which we were then classified using the *classify* Matlab command.

As a side note, included in the main script *SVDScript.m* was also numerous points of diagnostics. However, since printing to the screen is expensive these were designed with the ability to be suppressed.

To verify changes in accuracy due to method vs. chance, a different approach was taken for each of the three parts. For the first part, a smaller amount of training samples was used to see the robustness of the algorithm given a small N. In the second part three very similar sounding bands were chosen to make the task more difficult. In the final case a large amount of data was used and the genres were as distinct as possible to test the upper limits on the accuracy of classification.

4 Computational Results

5 Conclusion

A
[u,s,v] = svd(A, OPTION) Computes the singular value decomposition for the given matrix. In this report we use the option flag '0' to specify the reduced SVD.
classify Matlab function for conducting linear discriminant analysis on the rows of three passed arguments.
spectogram Matlab function for the short-time fourier tranform Used in this analysis for returning the one-sided fourier transform accross several points across a clip of the song

B

References