AMATH 482: HOMEWORK 2

TIANRUI ZHOU

AMath Department, University of Washington, Seattle, WA
tianrz5@uw.edu

Abstract. This report is about solving a Captured Motion Recognition problem of a humanoid robot. The content will be divided into five sections: Introduction and Overview, Theoretical Background, Algorithm Implementation and Development, Computational Results, and Summary and Conclusions.[1] They will elaborate on how I used Principal Component Analysis and Singular Value Decomposition to process the raw data given. I used Python code as my data processing tool and generated visual results with it. I will provide detailed explanations of my understanding of the mathematical foundations of the methods and algorithms. I will present my computational results and my conclusion for this project.

1. Introduction and Overview

I am working on *train* and *test* data from the file `hw2datanpy.zip`, which is from the humanoid robot OptimuS-VD. It has built-in sensors which record the movements of its 38 joints with rate of $60Hz$, and the data of the movements are recorded as Euler angles that further transformed to xyz coordinates. The dimension of the raw data is 114×100, which stands for 100 timesteps and 38 location for each of the xyz coordinate.[1]

My goal is using dimensions that are lower than the original number of coordinates to create projections of the recordings of the robot and visualize each movement (walking, jumping, running). First use the training data to design an algorithm that is capable of telling which movement OptimuS-VD is doing. Then, I will use the test data to test if my algorithm maintains in similar accuracy as that of the training data.[1]

2. Theoretical Background

My goal of the whole algorithm is using math foundation of Principal Component Analysis, Singular Value Decomposition, and Machine Learning to design a classifier for different kinds of movements, and employ it in truncated data. To begin with, I compile all the sample data, which has five samples for each of the three movements, in one matrix, and then it has dimension of 114×1500. In order to compute the PCA modes of the compiled data, we need to first center the data. To center the data, subtract the mean from the original data:

$$X_{\text{centered}} = X_{\text{train}} - X_{\text{train\_mean}}$$

The basis of PCA modes can be extracted through the columns of the left singular vector U matrix from SVD: $A = U\Sigma V^T$. The result of the singular value decomposition can be used to

Date: February 13, 2024.

calculate the cumulative energy:

$$E_i = \frac{\sum_{j=1}^{i} dS_j^2}{\sum_{j=1}^{n} dS_j^2}$$

It calculates the cumulative energy by dividing a certain proportion of the squared singular values by the total sum of squared singular values. dS is the singular value, j means the j th singular value, and E_i stands energy for mode i.

In order to plot the projected data in truncated PCA space for three different movements, we first need to project the data to PCA spaces. The projection of a given sample z onto k-PC modes is achieved by $z_k = U_k^T z$, where U_k^T is the transposed U matrix from SVD with first k column vectors and column vectors from $k+1$ and onward are set to be zero vectors. [1] In our case, we set $k = 2$ for the 2D 2-PCA plot, and set $k = 3$ for the 3D 3-PCA plot.

We use Machine Learning concept to test the accuracy of the classifier, which can classify each data to a certain movement. My ground truth labels of 0, 1, 2 stands for the actual category of walking, jumping and running. To design the classifier, we first calculate the centroids of the projected data for each movement in reduced k PCA spaces. By calculating and finding the minimal distance between a data point and those centroids, train labels can be created using labels 0, 1, 2. The accuracy can be computed by calculating the percentage of samples for which the ground truth and the trained labels match. Lastly, we use the well-designed classifier on the test data, and also compute the the percentage of samples for which the ground truth test label and the test labels match.[1]

3. Algorithm Implementation and Development

In the python code, I used some software packages to help me do the calculation and analysis.

The `NumPy`[2] library is used for numerical operations and array manipulations. Some built-in functions in this package are really crucial to my program. `np.load`[8] is used to load data from a file, and it specifically loads the recorded location data of the joints movements from a file; `np.linalg.svd`[7] is a function that is doing Singular Value Decomposition (SVD), and it returns three matrices U, Σ, and V^T from $A = U\Sigma V^T$; `np.cumsum`[5] returns the cumulative sum of elements along a given axis. `np.concatenate`[4] joins a sequence of arrays along an existing axis and returns the concatenated array. `np.repeat`[9] outputs an array that repeat each element of an input array after themselves. `np.zeros_like`[10] returns an array of zeros with the same shape and type as a given array. `np.linalg.norm`[6] calculates and returns the 2-norm of a given matrix or vector.

`sklearn.metrics`[11] module implements functions assessing prediction error for specific purposes, and the function `accuracy_score`[12] computes the accuracy classification score of the labels in machine learning.

The package `Matplotlib`[3] is used to make visualization of my data.

4. Computational Results

My computational result includes the plot of the first five PCA modes in xyz space, the cumulative energy of PCA modes, the 2D and 3D trajectories of projected X_{train} in truncated PCA space

for three different movements, and the numerical results as well as a plot for the accuracy of the classifier for both train and test data.[1]

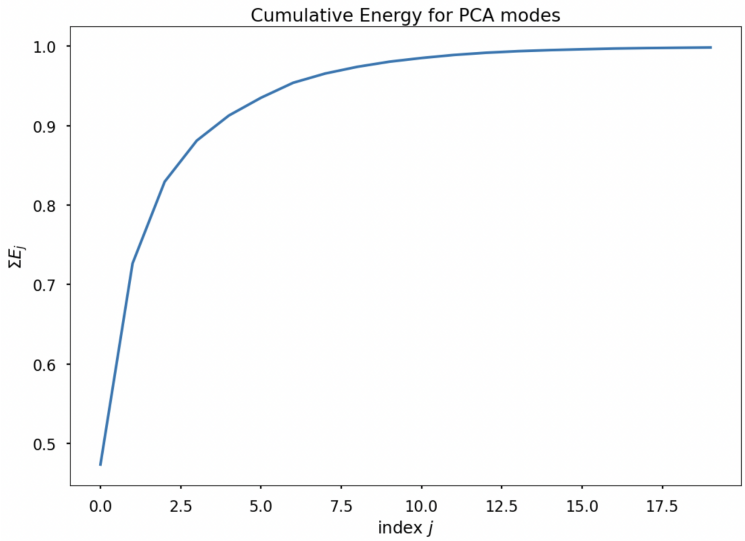


FIGURE 1. Cumulative Energy (Frobenius norm) for PCA modes

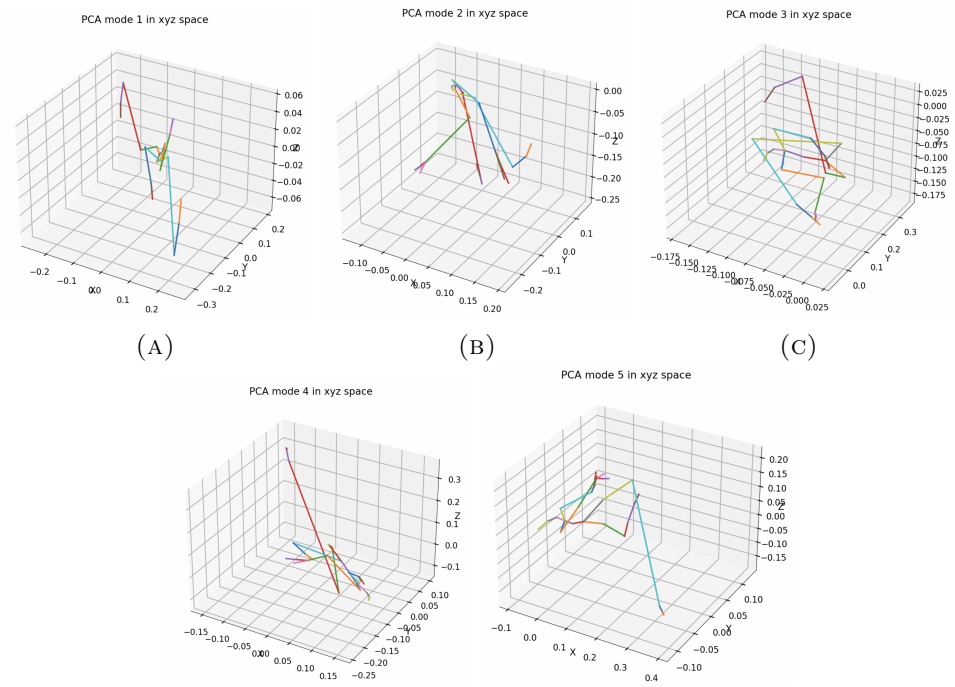
After compiling all the train samples into a matrix, Figure 1 plots the cumulative energy for different numbers of PCA modes. The x-axis stands for how many PCA modes are kept, the y-axis represents the corresponding cumulative energy (1.0 refers to 100%). As the number of PCA modes kept increases, we observed that the cumulative energy also increases.

According to the plot, for 70% of the energy, we only need about 2 PCA modes; for 80% of the energy, we need about 3 PCA modes; for 90% of the energy, we need about 5 PCA modes; for 95% of the energy, we need about 7 PCA modes. So, we can conclude that the first few PCA modes are already able to capture the majority of information of the data.

Then, I plot the first 5 PCA modes in xyz space. Figure 2 has five subplots (A B C D E), where each stands for one PC modes in the first five PCA modes. PCA modes indicate the "optimal" directions to represent the variance of the data and the first five capture the dominant movement pattern in X_{train} for the robot [13]. Through observing the lines with different colors, we can get to know the magnitude and direction of the variability of the movements for each joint.

I truncated the PCA modes to 2 and 3 modes, and plot the 2D trajectory and 3D trajectory in the truncated PCA spaces, as shown in Figure 3. In subfigure (A) 2D plot, the x-axis is PC1 and the y-axis is PC2, and each color stands for a certain movement from walking, jumping and running. For the subfigure (B) 3D plot, the third axis PC3 is added, and it can demonstrate the overall patter of the movements more clearly.

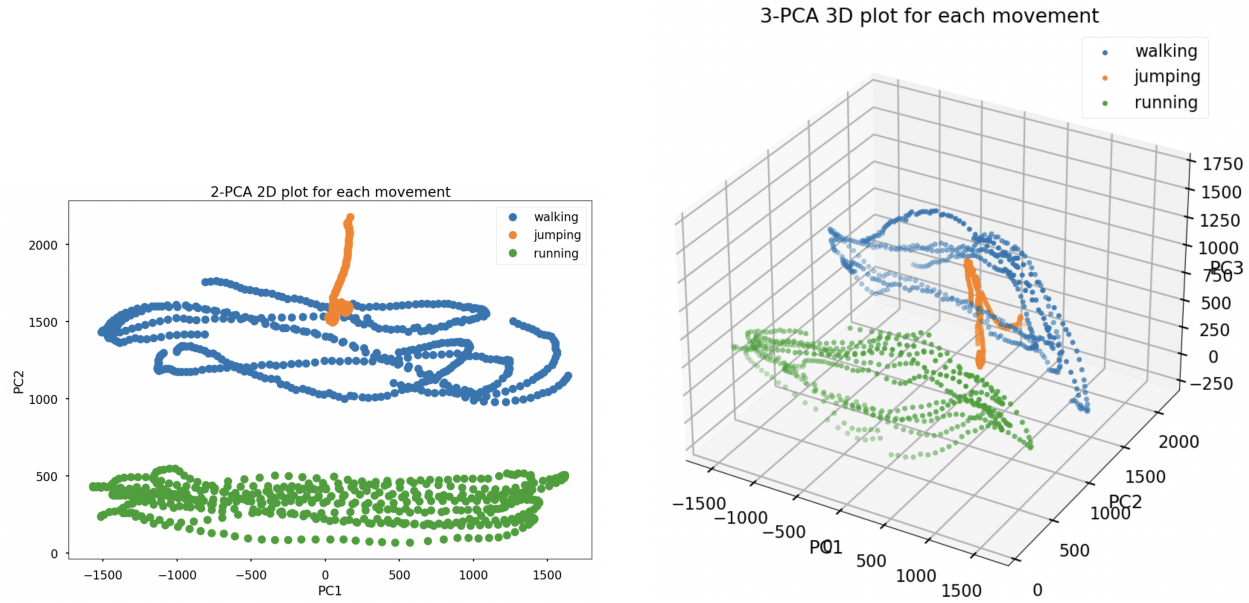
Through calculating the percentage of samples for which the ground truth and the trained labels match, we can get the curve of the accuracy of the classifier as a function of number (k) of PCA



(A) (B) (C)

(D) (E)

FIGURE 2. First 5 PCA modes in xyz space



(A) (B)

FIGURE 3. 2D and 3D trajectories of projected X_{train} in truncated PCA space for three different movements

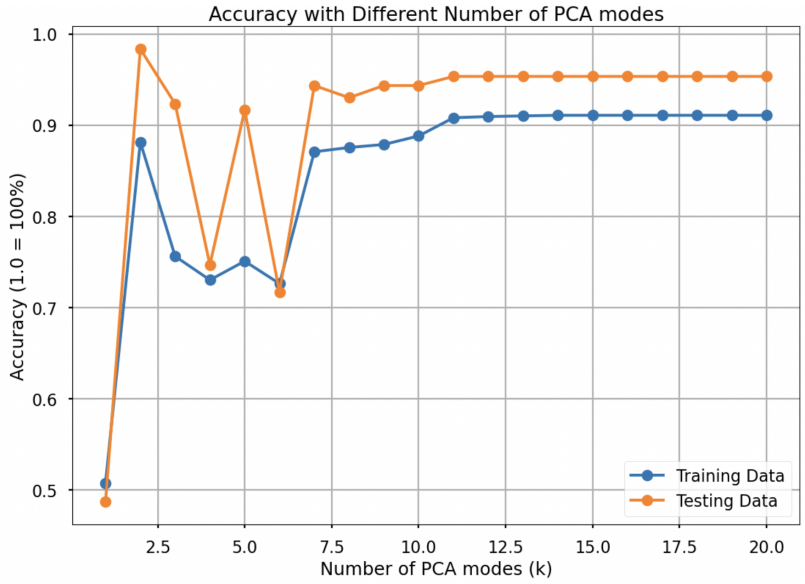


FIGURE 4. Comparison of the accuracy of the classifier for both train and test data

modes used by implementing the train data.
In Figure 4, the blue curve represents the accuracy of the classifier when using the train data. Through observation and computational result, we can conclude that when k is 14 or larger, the accuracy will keep in a stable value around 91%. The orange curve represents the accuracy of the classifier when using the test data. By computing, the accuracy of the classifier is stable when k is equal or greater than 11, and it gets to the accuracy of around 95.3%.
According the comparison plot in Figure 4, we can states that the overall pattern of the accuracy curve is the same for training data and test data. However, the accuracy in test data is slightly higher than in the training data for most of the k values.

5. Summary and Conclusions

This project gets raw data from the humanoid robot OptimuS-VD and uses lower dimensions to create projections of the recordings of the robot and visualize the trajectory in 2D and 3D plots. Through analyze the cumulative energy, we can conclude that we are able to capture majority of information only use the first five PCA modes. Using the concept of Machine Learning, we design a classifier using the train data to classify each data point to one of the movement categories: walking, jumping, and running. After we have the classifier well designed, it is implemented on a set of test sample, in order to see if the accuracy of the classifier maintains at a similar way as that of the train data.

In the future, this set of idea may be used to capture various other movements, for instance, in the programming of robot referees for Olympic diving competitions. It could be used to compare the standard movements in the air with the actions performed by athletes in real-time, generating scores accordingly.

Acknowledgements

The author is thankful to Prof. Eli Shlizerman for useful information posted on discussion board about the plotting the first five PCA modes in xyz space part for this project. We are also thankful to peer Ziyi Zhao and all the classmates on Discord for meaningful discussion of logic of the coding

part, and helping me debug my code successfully.

References

[1] Directions, reminders and policies. PDF, 2024. File: AMATH482582$_{H}$*omework*2*.pdf*.

[2] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020.

[3] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[4] NumPy Community. *NumPy np.concatenate Function*, 2022. https://numpy.org/doc/stable/reference /generated/numpy.concatenate.html.

[5] NumPy Community. *NumPy np.cumsum Function*, 2022. https://numpy.org/doc/stable/reference /generated/numpy.cumsum.html.

[6] NumPy Community. *NumPy np.linalg.norm Function*, 2022. https://numpy.org/doc/stable/reference /generated/numpy.linalg.norm.html.

[7] NumPy Community. *NumPy np.linalg.svd Function*, 2022. https://numpy.org/doc/stable/reference /generated/numpy.linalg.svd.html.

[8] NumPy Community. *NumPy np.load Function*, 2022. https://numpy.org/doc/stable/ reference/generated/numpy.load.html.

[9] NumPy Community. *NumPy np.repeat Function*, 2022. https://numpy.org/doc/stable/reference /generated/numpy.repeat.html.

[10] NumPy Community. *NumPy np.zeros_like Function*, 2022. https://numpy.org/doc/stable/reference /generated/numpy.zeros$_{l}$*ike.html*.

[11] scikit-learn Development Team. *scikit-learn*, 2022.

[12] scikit-learn Development Team. *scikit-learn accuracy_score Function*, 2022. https://scikit-learn.org/stable/modules/ generated/sklearn.metrics.accuracy$_{s}$*core.html*.

[13] E. Shlizerman. Amath 482 lecture 9 notes, 2023. University of Washington.