AMATH 482/582: HOMEWORK 2

HAI VAN LE

Applied Mathematics Department, University of Washington, Seattle, WA
levh@uw.edu

Abstract. For this project, we are interested in constructing a projection of the recordings of the movements of 38 joints of OptimuS-VD, a humanoid robot, to a lower dimension than the number of coordinates. Then, using that information, we will come up with an algorithm to identify the type of movement that the robot is performing in real life.

1. Introduction and Overview

We are given 15 datasets which are recordings of the movements of OptimuS-VD: walking, jumping, and running. The recordings were obtained from the sensors on the robot which capture the movements of 38 of its joints with a rate of 60 Hz in the form of Euler angles and can be projected to xyz coordinates. Each sample is recorded within 1.4 secs (100 timesteps) in the form of a 114x100 matrix. The first dimension is the recording of 38 locations of the robot's joints and the second is the recording of the timesteps. The task requires us to construct an algorithm that can tell what kind of movement the robot is making in real life. In order to test the accuracy of the algorithm, a test sample is provided.

2. Theoretical Background

This project mostly deploys Singular Value Decomposition (SVD) and Principal Component Analysis (PCA) [1], all of which are explained briefly below.

2.1. **Singular Value Decomposition.** By definition, a singular value decomposition is a factorization of a matrix into several constitutive components all of which have a specific meaning in applications. For any $A \in \mathbb{R}^{n\times m}$, there exist unitary matrices $U \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{m\times m}$ and a diagonal matrix $\Sigma \in \mathbb{R}^{n\times m}$ with positive entries such that $A = U\Sigma V^T$. In order to apply SVD to our given data, we need 3 transformations:

$$Ax = U\Sigma V^T x \Longrightarrow x \xrightarrow[\text{rotation}]{V^T} V^T x \xrightarrow[\text{scaling}]{\Sigma} \Sigma V^T x \xrightarrow[\text{change of basis principal components (PCA)}]{U} U\Sigma V^T x$$

2.2. **Principal Component Analysis.** This project represents a key application of the SVD, or alternatively a variant of Principal Component Analysis (PCA). PCA is a dimensionality reduction technique employed to discover patterns in high-dimensional data. It is used to transform the initial variables into a new set of variables called principal components, which are linear combinations of the initial variables. The order of the principal components is that the first component tells us the maximum variance in the data, the second component tells us the maximum remaining variance, etc. With these principal components, the dimension of the dataset can be reduced significantly yet still preserve most of its important information.
We can measure the variance of the data in each axis that is centralized with a mean of 0 by:
$C_x = cov(x) \sim \frac{1}{N-1} XX^T$: approximation of covariance matrix

Date: March 7, 2024.

$C_x \sim \frac{1}{N-1} U\Sigma V^T \cdot V\Sigma U^T = \frac{1}{N-1} U\Sigma^2 U^T$
PCA modes are eigenvectors of C_x and G^2 are eigenvectors of C_x. Thus, PCA modes indicate the optimal directions to represent variance of the data.

2.3. **a.**

3. Algorithm Implementation and Development

This project deploys different library interfaces such as *NumPy* to load the data, *matplotlib.pyplot* - a plotting library for the Python programming language and its numerical mathematics extension NumPy. Most importantly, it uses *sklearn* for PCA.
Specifically, the data is provided in two sets: training and testing. There are 15 datasets in the training part, five for each different movements including walking, jumping and running of the robot. The testing data folder has one dataset for each movement to test the accuracy of our model. After applying PCA, we obtain principal components, each of which represents a direction in the original feature space. When we visualize PCA, we will be able to understand how the original features contribute to each principal component. In this project, we are given with the skeleton data also visualized in the Helper file.
First step is to load the skeleton data from the training datasets which contain motion capture data representing different movements including walking, jumping, and running. Since each sample of data has 100 timesteps with a dimension of 114x100, when we stacked all training data into a single matrix using *hstack*, its dimension was 114x1500. To project the data onto the PC directions, we first centered the training data by subtracting the mean from each feature to center the data around the origin. Then, we performed PCA on the centered data to obtain the principal components and projected the centered data onto the principal components. Then, we used PCA to investigate the dimensionality of the data. By applying PCA to the stacked data, we obtained principal components representing the main directions of variation in the data. After that, we visualized the first 5 PCA modes as required. Each PC mode is a vector of U so we plotted it as a vector 114 points and their index. Next, we used Singular Value Decomposition (SVD) from the SVD notebook to perform dimensionality reduction and approximate the training data matrix X_{train}. Here, we calculated the cumulative energy percentage by first computing the explained variance ratio for each principal component using *explained variance ratio*. This ratio represents the proportion of the total variance explained by each mode. We then use *np.cumsum* to sum up the explained variance ratios of the retained principal components to obtain the cumulative explained variance ratio. To get the percentage, we simply multiplied the cumulative explained variance ratio by 100. Lastly, we evaluated the accuracy of the classifier on the provided test samples by projecting them onto the PCA space obtained from the training data and comparing the predicted labels with the ground truth labels. Specifically, for each k-modes PCA truncation, we computed the centroids for each movement in the reduced k-modes PCA space. The classifier then *assigned labels* to each sample based on the minimal distance, calculated using the function *distance* which computes the Euclidean distance between points in the PCA space, to the centroids. Then, we computed the accuracy of the trained classifier. That concludes the Classification Algorithm.

4. Computational Results

The first five PCA modes are shown in 1. The cumulative energy is plotted in 2.
For task 2, I found that:

- To approximate X_{train} up to 70.0% of its energy, you need 2 SVD modes.
- To approximate X_{train} up to 80.0% of its energy, you need 3 SVD modes.
- To approximate X_{train} up to 90.0% of its energy, you need 5 SVD modes.
- To approximate X_{train} up to 95.0% of its energy, you need 7 SVD modes.

For task 3, as we can see from the 2D 3 and 3D 4 trajectories. The jumping is more visibly self-explanatory. The density of running also makes sense since the the movement is faster than walking.
For task 4, I found that the centroid for walking is [-36.88211143 253.35282541 175.91202104], for jumping is [-23.88986635 499.36826149 -72.5000755] and for running is [60.77197779 -752.7210869 -103.41194553].
For task 5, training accuracy at $k = 2$ is 88.13%, at $k = 3$ is 75.6%, at $k = 4$ is 73%, at $k = 5$ is 75.07%, at $k = 6$ is 72.6%, at $k = 7$ is 87.07%, at $k = 8$ is 87.54%, at $k = 9$ is 87.87%, at $k = 10$ is 88.8%, at $k = 11$ is 90.8%, at $k = 12$ is 90.93%, at $k = 13$ is 91%, at $k = 14$ to $k = 20, k = 30$ is 91.07%. Thus, for optimal accuracy, k should be at least 14. Overall, the training accuracy steadily increases as k grows initially. At $k = 2$, the training accuracy is relatively high at 88.13%, indicating that the classifier captures a significant amount of variance with just two principal components. The training accuracy fluctuates slightly as k increases. At $k = 14$ to $k = 20, 30$, the training accuracy stabilizes around 91.07%, indicating that adding more principal components beyond $k = 14$ doesn't significantly improve the model's ability to represent the training data.
For task 6, testing accuracy at $k = 2$ is 98.33%, at $k = 3$ is 92.33%, at $k = 4$ is 74.67%, at $k = 5$ is 91.67%, at $k = 6$ is 71.67%, at $k = 7$ is 94.33%, at $k = 8$ is 93%, at $k = 9$ is 94.33%, at $k = 10$ is 94.33%, at $k = 11$ to $k = 20, k = 30$ is 95.33%. Comparing these testing accuracy scores with the training accuracy scores, we can see that the testing accuracy score is highest when $k = 2$ and lowest at $k = 6$. The testing accuracy fluctuates more than the training accuracy as k increases. At $k = 2$, the testing accuracy is highest at 98.33%, indicating good generalization performance with just two principal components. The testing accuracy drops significantly at $k = 3$, indicating overfitting or loss of important information as more principal components are added. The testing accuracy is stabilized at the range $k = 11$ to $k = 20$, indicating that the model's generalization performance is optimal within this range and adding more principal components doesn't improve generalization performance further. While the training accuracy continues to increase with k, it may not reflect the model's true generalization performance, as indicated by the fluctuations and eventual plateauing of testing accuracy. We recommend considering both training and testing accuracy to determine the optimal k for PCA truncation. In this case, any k from 11 to 30 seems to provide a good balance between capturing variance in the training data and generalizing well to unseen test data.

5. Summary and Conclusions

This project attempted to build a projection of the recordings of the movements of 38 joints of OptimuS-VD, a humanoid robot, to a lower dimension than the number of coordinates. Then, we showed an algorithm to identify the type of movement that the robot is performing in real life. This project made use of PCA and SVD technique and libraries such as *sklearn* to predict accuracy score of the model.

Acknowledgements

The author is thankful to Prof. Eli for useful discussions about the SVD algorithm and Python libraries. She is also thankful to the TAs who answered all questions regarding class materials and resources.

References

[1] J. Kutz. *Methods for Integrating Dynamics of Complex Systems and Big Data*. Oxford University Press, Oxford, 2013.

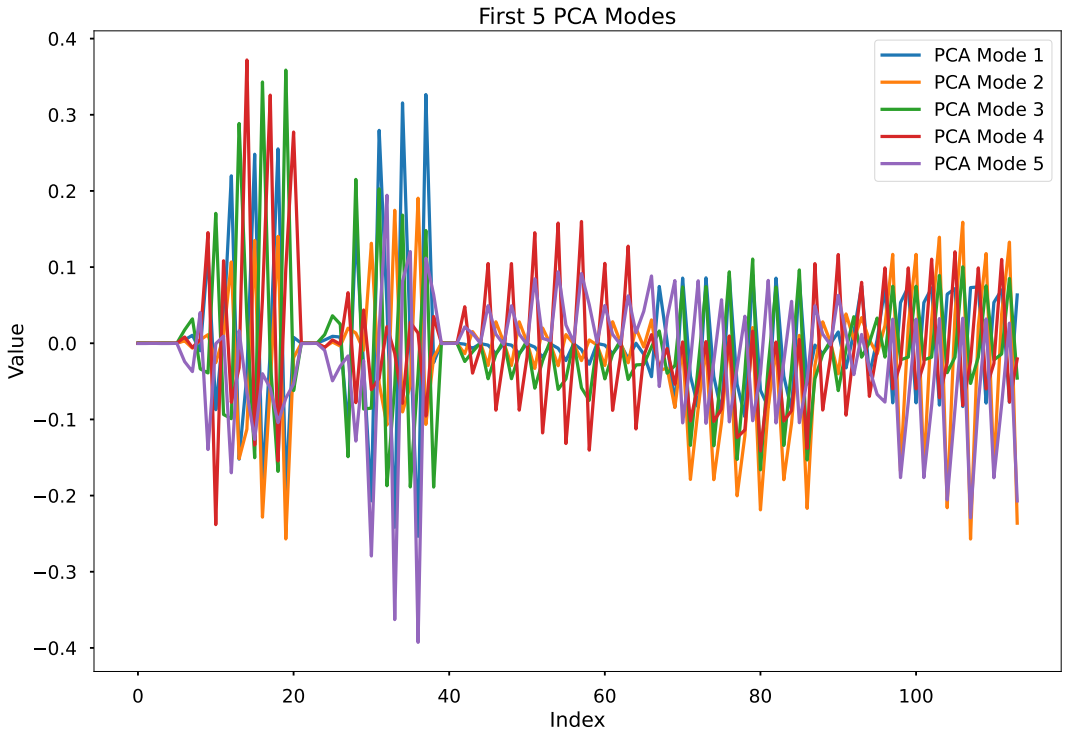

FIGURE 1. PCA modes as a a vector of 114 points and their index

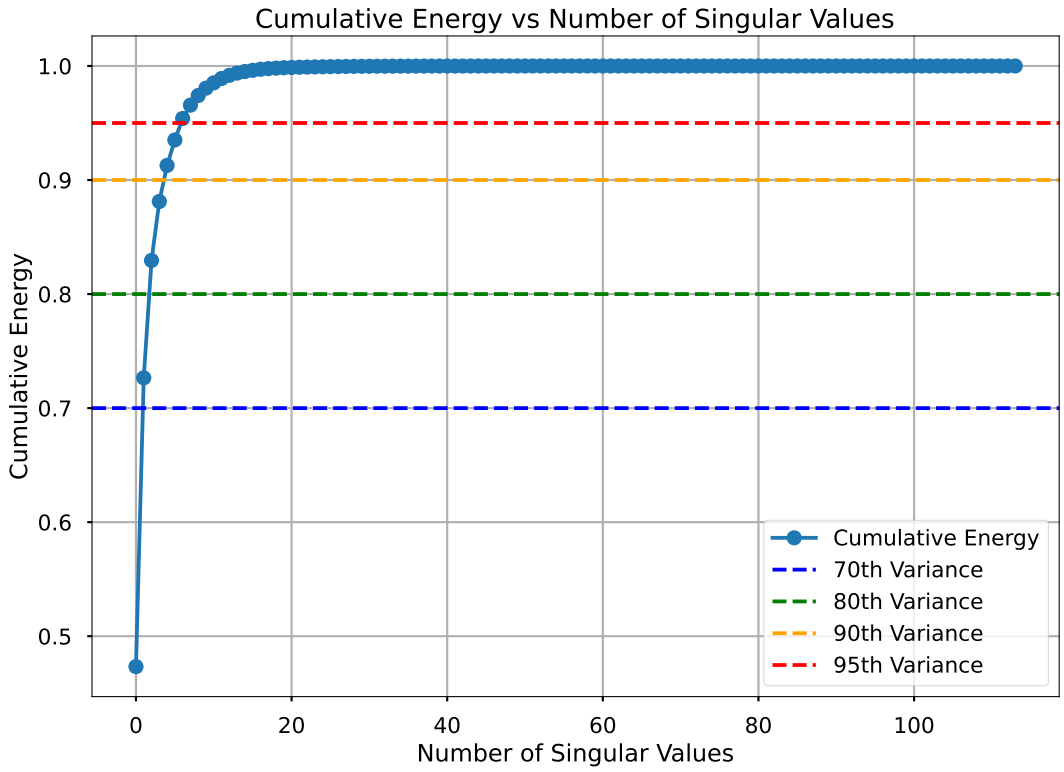

FIGURE 2. PCA energy vs Number of Modes k

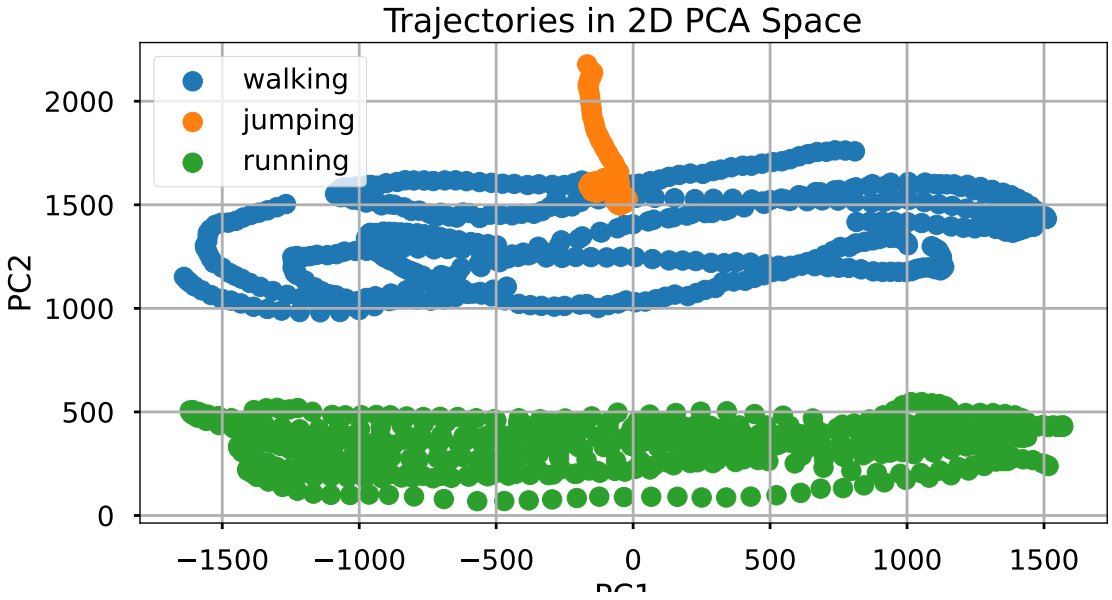

FIGURE 3. Projected X_{train} in 2D trajectories

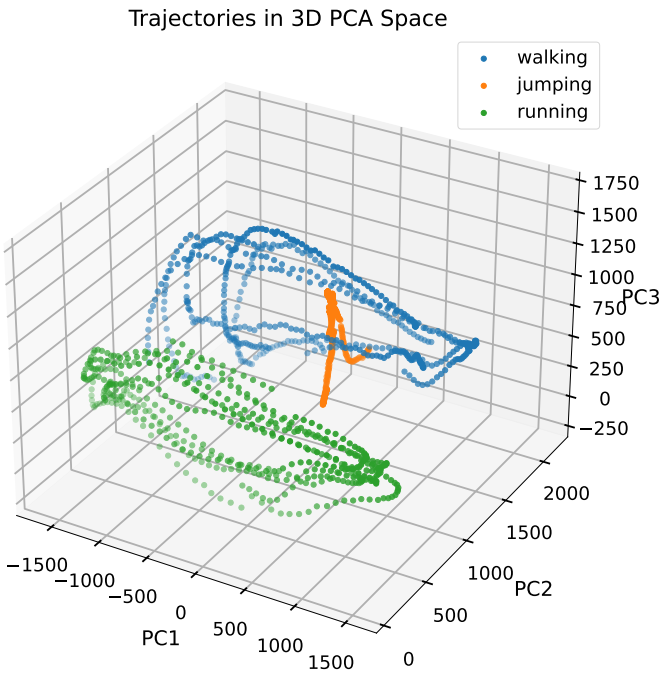

FIGURE 4. Projected X_{train} in 3D trajectories