Amath482HW2

Tong Wu

February 2025

1 Introduction

The humanoid robot OptimusVD has been recorded for its movement of 38 joints and tracked with a rate of 60Hz. This report will build a projection of the recordings to lower dimensions visualize the movement and design the algorithm that can recognize the movement OptimusVD will perform in real time. This report uses Principal Component Analysis (PCA) to lower the dimension and find singular values, explained variance, and cumulative energy (cumulative sum of eigenvalues). Throughout truncating data into 2D and 3D, labeling each sample, and computing the accuracy of the trained classifier, this report visualizes robotic movement data and predicts the labels for the test data.

2 Theoretical Background

2.1 Singular Value Decomposition, Eigenvalues and vectors

Principal Component Analysis (PCA) is the fundamental technique for analyzing data, and Singular Value Decomposition (SVD) decomposes the matrix into left singular values(U), and singular values(Σ), which correspond to the square root of eigenvalues and right singular values (V). To compute SVD:

$$A^TA = (U\Sigma V^*)^T(U\Sigma V^*) = V\Sigma U^*U\Sigma V^* = V\Sigma^2V^* \quad [1.a]$$

$$AA^T = (U\Sigma V^*)(U\Sigma V^*)^T = U\Sigma V^*V\Sigma U^* = U\Sigma^2U^* \quad [1.b]$$

2.2 Principal Component Analysis

Principal Component Analysis is one of the techniques that can take high-dimensional data and present it in a lower-dimensional form without losing too much information. By transforming correlated variables into linearly uncorrelated components and diagonalizing the covariance matrix using SVD, PCA isolates the most significant modes of variation[2].

2.3 Explained Variance and Cumulative Energy

Explained Variance is a statistical measure that quantifies how the variance in a dataset is distributed among the eigenvectors obtained through Principal

Component Analysis (PCA). It is represented as the ratio of each eigenvalue to the sum of all eigenvalues[3]. This can be expressed in the code using the PCA function:

$$ExplainedVarianceRatio = \texttt{pca.explained\_variance\_ratio\_}$$

$$CumulativeEnergy = \texttt{np.cumsum(exp\_var)}$$

3 Algorithm Implementation and Development

Part 1:

- Load the data from the training folder, make a nested loop for three movements from number 1 to 5, and load these npy files into *numpy* array.

- From *sklearn* packages to import PCA, which reduces the dimension for the dataset. By transforming the training dataset Xtrain and fitting the data by *pca.fit*, it can capture the most significant variance. The equation which represents the transformation of the training data into a lower-dimensional space using Principal Component Analysis (PCA) is:

$$X_k = U_k^T X$$

- Find the singular values, explained values, and cumulative energy. Approximate Xtrain up to determine the minimum number of principal components (modes) needed to capture a given cumulative energy threshold. Using the `np.searchsorted`

  Part 2:

- Truncate the dataset to 2D and 3D modes with shapes of (1500,2) and (1500,3).
- Because we put 15 movements into the same file, we now have 15 * 100 columns in total, then we define the range of each movement. walking is 0 - 499, jumping is 500-999, and running is 1000-1500.
- Plot the data into 2 PC modes graph and 3 PC modes graph, with each axis representing each PC. Part 3:
- Import *accuracy_score* from *sklearn* package in order to compute the accuracy score.
- Assign the labels for each movement: (0–walking; 1–jumping; 2–running).

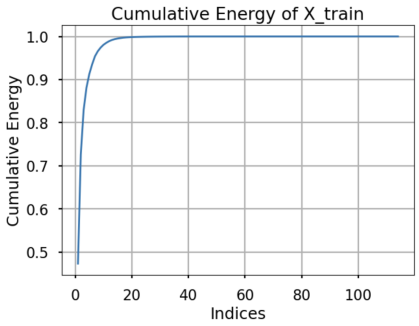

Figure 1: Cumulative Energy of X train dataset

- Define the classification function, using *np.linalg.norm* to compute Euclidean distance
- To compute the labels for various k values, we need to write a loop for each k value and truncate the training dataset to the first k principal components.
- Compute the centroids by taking the mean of the data across the time.
- We write a loop over each data to make each data labeled based on its distance to centroids.
- Use the accuracy_score to calculate the accuracy score for each k value. Part 4:
- Do the same task for testing data and plot the accuracy score comparison between the testing values and training values.

4 Computational Results

1. By applying **Principal Component Analysis (PCA)**, the data is projected onto a lower-dimensional space while preserving the most significant variance. Using `np.searchsorted()`, different thresholds are settled to retain **70%, 80%, 90%, and 95%** of the total energy. This for-loop finds the smallest number of principal components (modes).
   The results are:
   To capture 70% of the total energy, we need 2 modes.
   To capture 80% of the total energy, we need 3 modes.
   To capture 90% of the total energy, we need 5 modes.
   To capture 95% of the total energy, we need 7 modes.
   The results are illustrated in **Figure 1**.

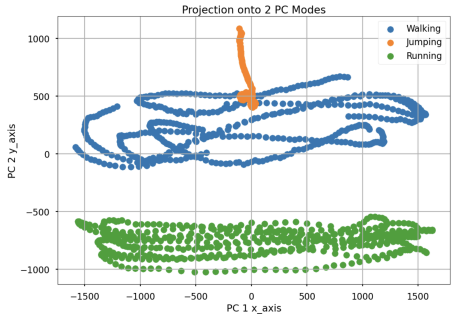

Figure 2: Projection onto 2PC modes - 2D

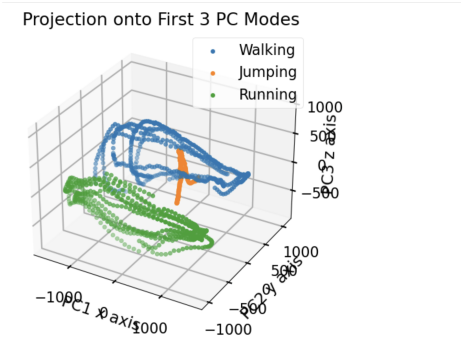

Figure 3: Projection onto 3PC modes -3D

2. By slicing datasets for each movement and projecting the 0th column for PC 1, the 1st column for PC 2, and the 2nd column for PC 3 (3D plot). These figures are Figure 2 and Figure 3.

3. Compute the accuracy rate for the trained labeling each k value, we can discover that:
   k = 1, Training Accuracy: 50.73%
   k = 5, Training Accuracy: 75.07%
   k = 7, Training Accuracy: 87.07%
   k = 10, Training Accuracy: 88.80%
   k = 20, Training Accuracy: 91.07%
   k = 50, Training Accuracy: 91.07%
   As the k values increase, which are kth principal components of truncation, the accuracy scores increase as well, but training accuracies stop increasing at 91.07%.

4. Assign the same method to the testing data, and the results are shown:

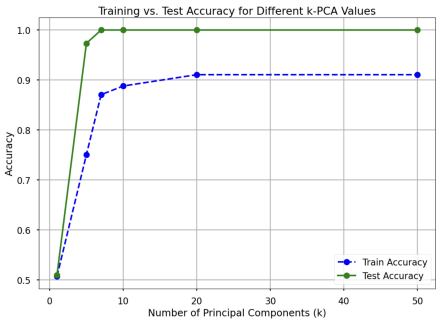

Figure 4: Training VS Testing Accuracy

k = 1, Training Accuracy: 51.00%
k = 5, Training Accuracy: 97.33%
k = 7, Training Accuracy: 100.00%
k = 10, Training Accuracy: 100.00%
k = 20, Training Accuracy: 100.00%
k = 50, Training Accuracy: 100.00%
The comparison are shown on Figure 4:

5 Summary and Conclusions

By applying **Principal Component Analysis (PCA)**, the accuracy of the trained classifier is calculated over different numbers of principal components (**k values**). The training accuracy starts at **50.73% for k = 1**, increasing to **91.07% at k = 50**. Testing accuracy, however, reaches **100% at k = 7**, suggesting that less data (300 columns) might more possibly reach 100% accuracy rate than more data (1500 columns). This report highlights the effectiveness of PCA in labeling and testing for accuracy.

6 Acknowledgments

Thanks for Rohin Gilman, the Teaching Assistant for supporting and helping.

7 References

References

[1] Kutz, J. N. (2013). *Data-driven modeling & scientific computation: Methods for integrating dynamics of complex systems and big data*. Oxford University Press.

[2] Shalizi, C. R. (2012). *Advanced data analysis from an elementary point of view* (Lecture 18). Retrieved from https://www.stat.cmu.edu/ cshalizi/uADA/12/lectures/ch18.pdf

[3] Vitalflux. (n.d.). *PCA Explained Variance Concept & Python Example*. Retrieved from https://vitalflux.com/pca-explained-variance-concept-python-example/