AMATH 482: HOMEWORK 2

NICHOLAS NUGRAHA

Applied Math Department, University of Washington, Seattle, WA
nickjn@uw.edu

Abstract. This report applies Principal Component Analysis (PCA) to classify robotic movements recorded from a robot's joint sensors over time. Using PCA, the dataset is reduced while also preserving key movement patterns. Projecting the movements into a lower-dimensional PCA space allows for classification via nearest-centroid analysis. Results show that a range from $k \in [10, 20]$ provides the best balance with train and test accuracy being over 90%.

1. Introduction and Overview

In this assignment, I work with data from a humanoid robot called OptimuS-VD which has built in sensors that record the movements of its 38 joints with a rate of 60Hz. The joint movements are recorded as Euler angles that can be transformed into xyz coordinates. There are 5 samples for each of the 3 movements that OptimuS-VD knows how to do: walking, jumping, and running. The samples are recorded for 100 timesteps and is a matrix of 114x110, where the first dimension records $x_1, \ldots, x_{38}, y_1, \ldots, y_{38}, z_1, \ldots, z_{38}$ locations of the joints and the second dimensions are the timesteps. The main goal is to build a projection of the recording to a lower dimension than the number of coordinates, visualize the movements and then design an algorithm that can recognize which action OptimuS-VD is performing.

2. Theoretical Background

Since we are in the context of dimensionality reduction, there are two mathematical concepts we must first understand: Singular Value Decomposition (SVD) and Principal Component Analysis (PCA). With these tools, we can transform high-dimensional data like our robot data into a more compact and interpretable form. The data we have contain redundant features that make it difficult to interpret so reducing the dimensionality retains the most significant structures, essentially ignoring the redundant information.

SVD is a matrix factorization technique that decomposes any $m \times n$ matrix A into three matrices:

$$A = U\Sigma V^T$$

where U is an $m \times m$ orthogonal matrix containing left singular vectors, Σ is an $m \times n$ diagonal matrix with singular values arranged in descending order, and V^T is an $n \times n$ orthogonal matrix containing right singular values. The singular values in Σ indicate the significance of each corresponding dimension, with large values representing more important features in the data.

PCA is a statistical method used for transforming high-dimensional data into lower-dimensional space, preserving variance as much as possible. This method identifies the directions in the data that exhibit the greatest variability. These are known as principal components which capture the most significant patterns in the data. PCA is also often performed using SVD. Our dataset X is decomposed into $X = U\Sigma V^T$ where the columns of V correspond to the principal components.

Date: February 15, 2025.

The number of principal components k we choose also has an effect on how much information was retained in the process, as well as how well classification tasks, like identifying the movement, can be performed.

3. Algorithm Implementation and Development

Using `sklearn`'s implementation of PCA, we can easily break down our matrix into components that we can then analyze. In this PCA algorithm, the data is automatically centered, however we do have to transpose our matrix to abide with the row and column conventions of `sklearn`. Along with this, we use `numpy`'s tools to perform various mathematical operations and `matplotlib` to visualize our findings.

4. Computational Results

In order to see how much variance in the data is preserved as we increase the number of principal components, we can look at the cumulative energy which tells us how many components are required to retain a certain percentage of the original dataset's variance.

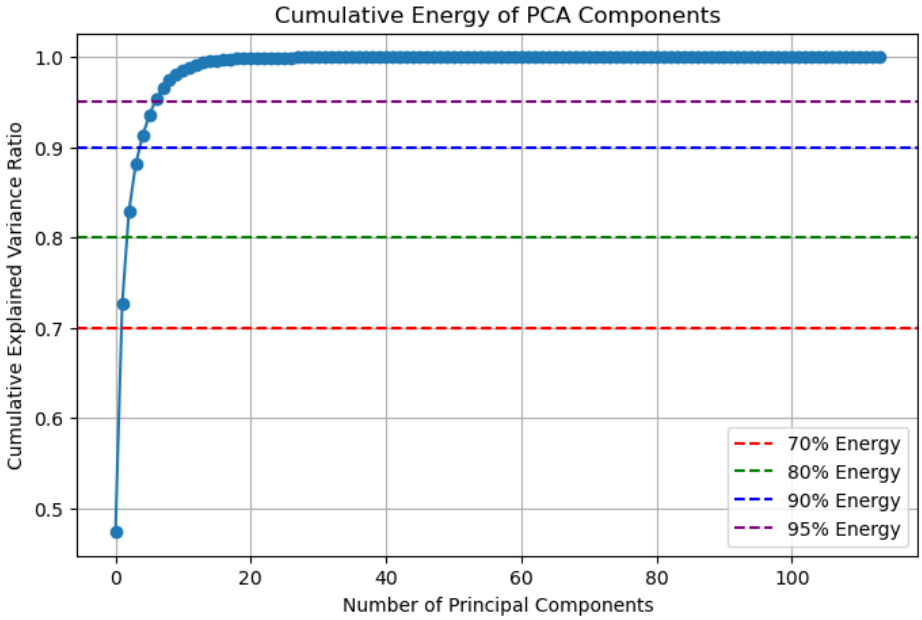
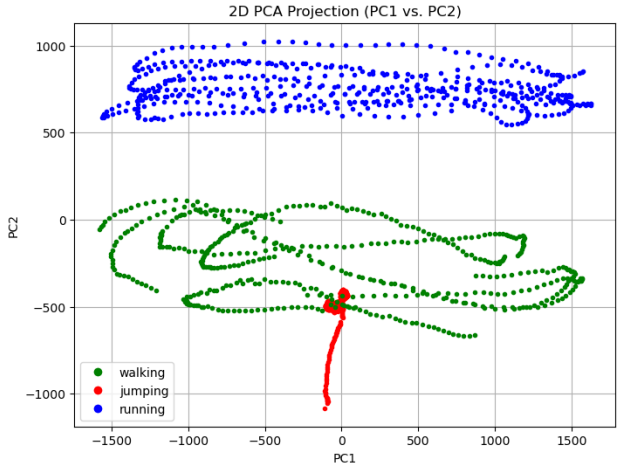


Figure 1. Cumulative energy based on how many principal components are chosen

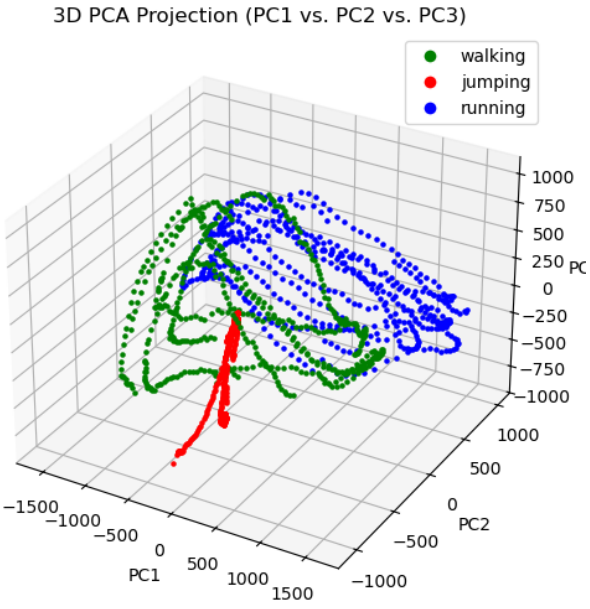
Looking at Figure 1, the smaller the number of components, the less cumulative energy there is. This means the data is very compressed, but not detailed, leading to problems later down the line when we perform classification. Increasing the number of components leads to higher cumulative energy, so more movement details are retained at the cost of computational complexity

We want to choose an optimal value for k in order to balance accuracy and efficiency. If we pick too few components the movement patterns will not be well-distinguishable, but if we pick too many we retain unnecessary details which increases computation time without improving the classification by a noticeable amount.

In order to visualize the movements of the robot, we choose a k-value of 2 and 3 (2 and 3 principal components) and plot the data in the 2D and 3D space, respectively.



(A) Movement in 2D space (PC1, PC2 coordinates)



(B) Movement in 3D space (PC1, PC2, PC3 coordinates)

FIGURE 2. Robot movement in 2D and 3D space

Figure 2a shows the movement in 2D space. Here, it is quite easy to distinguish between the different movements. Specifically, walking and running have similar patterns since they both are full body movements, whereas jumping forms more of a vertical pattern since the robot is only moving in the up and down direction. In Figure 2b, we can see the movement in the 3D space, which show more separation between the cluster of movements. Adding the third principal component adds information that was lost in 2D, making the classification more effective.

In order to actually classify the training and test sets, we establish a vector of ground truth labels with an integer per movement. For each movement, we then compute its centroid (mean) in k-modes PCA space. By calculating the minimum distance between the centroid and the projected point in the data set, we can classify which class the sample belongs in (walking, jumping, or running).

TABLE 1. Training Labels

k	Accuracy
2	0.8813
3	0.7560
5	0.7507
10	0.8880
20	0.9107
114	0.9107

TABLE 2. Test Labels

k	Accuracy
2	0.9833
3	0.9233
5	0.9167
10	0.9433
20	0.9533
114	0.9533

Tables 1 and 2 show the accuracy of the trained and test classifiers (percentage of samples for which the ground truth and the trained/test labels match). Looking at Table 1, the accuracy fluctuates a decent amount with lower values of k. However, the highest accuracy comes from when $k > 20$. After $k = 20$ there is no longer a change in accuracy, so choosing a higher dimension than 20 would not be useful and instead introduces redundant information rather than useful distinctions.

In Table 2 we classify on a test dataset that the robot has not seen. Overall, all accuracies at any value of k in the test set are much higher than in the training, suggesting that the model generalizes well. Here the optimal k-value is more difficult to determine. Since we want to have a high accuracy, while still maintaining a reasonable number of dimensions, $k = 10$ is the optimal value in this case. The accuracy at $k = 10$ is 94.3%. When going beyond $k = 20$ the accuracy stabilizes at 95.3% so it's not worth it to add the extra dimensions and potentially over complicating the data.

5. Summary and Conclusions

This study investigated PCA-based dimensionality reduction for robotic movement classification, using joint sensor data from walking, jumping, and running movements. SVD was used to extract the principal components and the movement data was projected into a lower-dimensional PCA space where classification was performed using distances from the centroids.

The results highlight that the number of principal components chosen has a large effect on how accurate the classifier predicts the movements. Overall, we found that values from $k \in [10, 20]$ yielded the best results, consistently scoring an accuracy of over 90%. There is some caution to be used when choosing k values as too small of a value can result in a loss of important information and too large of a value can introduce redundant information.

Overall, PCA successfully reduces the dimensions of the dataset and predicts the movements fairly accurately. Future work could explore alternative classification methods in PCA space.

Acknowledgements

The author is thankful to Professor Natalie Frank for useful lectures about PCA. We are also thankful to TA Rohin Gilman for assisting in office hours. We also acknowledge that artificial intelligence was used to make edits to this document, fix minor bugs, and gain a better understanding of the concepts.

References

[1] T. N. Community. *NumPy Documentation*, 2025. Accessed: 2025-01-25.
[2] N. Frank. Amath 482: Computational methods for data analysis. Lecture notes, University of Washington, 2025.
[3] J. N. Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. OUP Oxford, 2013.
[4] OpenAI. Chatgpt. `https://chat.openai.com/`, 2025.