AMATH 482/582: HOME WORK 2

REBECCA WANG

Amath Department, University of Washington, Seattle, WA
lufanw@uw.edu

Abstract. This project explores the use of Principal Component Analysis (PCA) for dimensionality reduction in motion-capture data from a humanoid robot, OptimuS-VD. The dataset records joint movements during different activities, including walking, jumping, and running. The goal is to reduce dimensionality while preserving essential motion characteristics and develop a centroid-based classifier for movement recognition.

1. Introduction and Overview

In this project, we analyze motion-capture data from a humanoid robot, OptimuS-VD, which logs 38 joints over 100 time steps (1.4 seconds). Three movements are recorded: walking, jumping, and running. Our goal is to reduce these high-dimensional signals via Principal Component Analysis (PCA) and then build a classifier to distinguish the movements.

We first compile all training data into a matrix, apply PCA, and determine how many principal components capture specified energy thresholds (70%, 80%, 90%, and 95%). Next, we visualize the reduced-dimensional trajectories in 2D and 3D by truncating to the first two or three PCA components. For classification, we label each frame by its distance to the centroids of each movement class in the k-dimensional PCA space. We measure accuracy on both training and test data for various values of k. Finally, we explore a kernel-based alternative (k-PCA) and compare its performance to standard PCA.

2. Theoretical Background

2.1. Principal Component Analysis (PCA). PCA is a dimensionality reduction technique used to extract dominant motion patterns in the dataset. It projects high-dimensional data into a lower-dimensional space by finding new orthogonal axes (principal components) that maximize variance. Given a data matrix X, the covariance matrix is defined as:

$$C_x \approx \frac{1}{N-1} X X^T. \tag{1}$$

Eigenvectors of C_x define the principal components, and the corresponding eigenvalues determine the variance along each component. The optimal number of components k is selected to retain a sufficient percentage of variance (e.g., 70%, 80%, 90%, 95%).

Date: February 15, 2025.

2.2. Singular Value Decomposition (SVD). SVD provides a mathematical foundation for PCA. For any matrix $A \in \mathbb{R}^{n\times m}$, SVD decomposes it into three matrices:

$$A = U\Sigma V^T \tag{2}$$

where:

- U and V are orthonormal matrices representing rotations and reflections.
- Σ is a diagonal matrix containing singular values, which represent scaling factors.

The PCA transformation corresponds to computing the projection:

$$Ax = U\Sigma V^T x. \tag{3}$$

This sequence involves three transformations: rotation (V^T), scaling (Σ), and another rotation (U). The columns of U are called **Principal Component Modes**, which serve as a change-of-basis matrix.

2.3. Explained Variance and Energy Thresholds. Principal components capture variance based on their eigenvalues λ_i. The explained variance ratio and cumulative variance are given by:

$$\frac{\lambda_i}{\sum_{j=1}^{n}\lambda_j}, \quad E_k = \sum_{i=1}^{k}\frac{\lambda_i}{\sum_{j=1}^{n}\lambda_j} \tag{4}$$

Additionally, energy retention in Singular Value Decomposition (SVD) is defined as:

$$\frac{\sum_{i=1}^{k}\sigma_i^2}{\sum_{i=1}^{\min(n,m)}\sigma_i^2}, \tag{5}$$

These metrics quantify how much information each component retains. Energy thresholds (e.g., 70%, 80%, 90%, 95%) help determine the optimal k to balance variance preservation and computational efficiency. Lower k reduces noise but may lose critical motion details, while higher k retains more variance but risks overfitting. The cumulative energy plot helps select the optimal k for effective motion representation.

2.4. Truncated Projection in PCA. In practical applications, it is computationally efficient to **truncate** PCA modes by selecting only the first k components. This is done by taking the first k columns of U:

$$U_k^T X = \tilde{X}_{PCA_k}. \tag{6}$$

This reduces dimensionality while preserving significant motion information. The optimal value of k is chosen based on cumulative variance analysis.

2.5. Motion Classification via Nearest Centroid. After reducing dimensionality, each motion (walking, jumping, running) is represented in the PCA space. We assign **ground truth labels** to each movement and compute **centroids** (mean feature vectors) for each class in the k-dimensional space:

$$c_i = \frac{1}{N_i}\sum_{j=1}^{N_i} X_j, \tag{7}$$

where c_i is the centroid of class i. New samples are classified by **nearest centroid assignment**, where a test sample is assigned the label of the closest centroid using Euclidean distance.

2.6. Kernel PCA for Nonlinear Classification. To handle nonlinear relationships in motion data, we implement **Kernel PCA (k-PCA)**, which extends PCA using kernel methods. Instead of performing PCA in the original space, we apply a kernel function $\phi(x)$ that maps data into a high-dimensional space where linear PCA is applied. This allows for capturing complex motion dynamics. We experiment with different kernels such as:

- **Radial Basis Function (RBF)**: Captures localized variance.
- **Polynomial**: Models polynomial interactions between joints.
- **Sigmoid**: A hyperbolic tangent-based transformation.

By comparing **linear PCA** and **k-PCA**, we assess whether nonlinear feature extraction improves motion classification accuracy.

3. Algorithm Implementation and Development

The implementation of this project leverages the following Python packages for efficient computation and visualization:

- **NumPy** (`numpy`): Used for numerical computations, matrix operations, and handling large datasets efficiently.
- **Matplotlib** (`matplotlib.pyplot`): Enables visualization of 2D and 3D data.
- **MPL Toolkits** (`mpl_toolkits.mplot3d`): A Matplotlib extension for rendering and interacting with 3D plots, essential for visualizing PCA projections.
- **scikit-learn** (`sklearn.decomposition, sklearn.metrics`): Provides tools for PCA, Kernel PCA, and classification accuracy evaluation.
- **glob & os**: Used for handling file directories and loading datasets from structured folders.

3.1. Data Loading and Preprocessing. The dataset consists of motion-capture sequences in `.npy` format, with each file containing 38 joint coordinates over 100 time steps. Training data is reshaped and concatenated into a single matrix to ensure consistency.

3.2. Principal Component Analysis (PCA). To reduce the dimensionality of the motion data, PCA is applied by first flattening the dataset into a two-dimensional representation. The principal components are extracted using Singular Value Decomposition (SVD), where the variance explained by each component is analyzed. The number of components k is chosen based on cumulative variance, retaining 70%, 80%, 90%, and 95% of the data's variance.

3.3. Motion Visualization and Projection. The PCA transformation is truncated to two and three dimensions to visualize the movement patterns. The first two principal components are used for a 2D scatter plot, and the first three components are used for a 3D visualization. These projections help analyze the separability of different movement types and observe whether they form distinct clusters.

3.4. Centroid-based Classification. Each movement type is assigned a unique integer label: walking (0), jumping (1), and running (2). The mean feature representation (centroid) for each class is computed in the k-dimensional PCA space. A new sample is classified by computing its Euclidean distance to each centroid and assigning it the label of the nearest centroid.

3.5. Evaluating Classification Performance. The classifier's accuracy is evaluated by varying the number of principal components from $k = 1$ to $k = 7$. The classification performance is assessed by computing the accuracy score, which measures the percentage of correctly classified samples. The results indicate the optimal k value, balancing dimensionality reduction and classification accuracy.

3.6. Testing on Unseen Data. The trained classifier is applied to test samples, which are projected onto the same k-dimensional PCA space. The test samples are classified using the nearest centroid method, and accuracy is compared to the training set. This step ensures that the model generalizes well to unseen motion sequences.

3.7. Kernel PCA for Nonlinear Classification. To explore whether a nonlinear transformation improves classification, we apply Kernel PCA (k-PCA) using different kernel functions:

- **Radial Basis Function (RBF)**: Captures local structures.
- **Polynomial**: Models polynomial interactions.
- **Sigmoid**: Mimics neural network activation functions.

Classification accuracy using k-PCA is compared to standard PCA across $k = 2, 3, 4, 5$ to determine its effectiveness. The results indicate whether k-PCA provides a better representation of motion patterns compared to standard PCA.

4. Computational Results

4.1. Dimensionality Reduction Using PCA. PCA was applied to extract **PCA spatial modes**, capturing dominant motion patterns. The **cumulative energy distribution** determined the number of components needed to retain key information. The dataset, reshaped from (38, 3, 1500) to (1500, 114), showed that 2, 3, 5, and 7 components retain 70%, 80%, 90%, and 95% of the total energy, enabling effective dimensionality reduction.

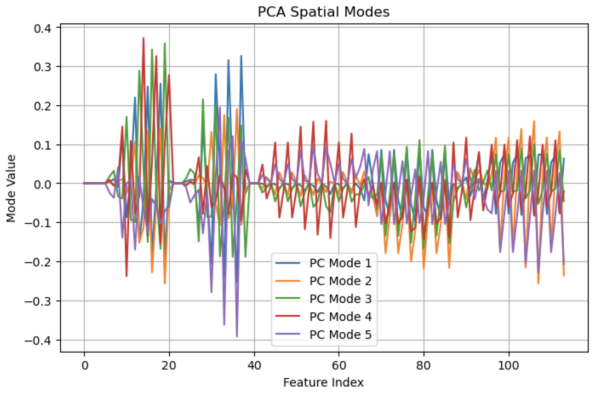


Figure 1. PCA Spatial Modes

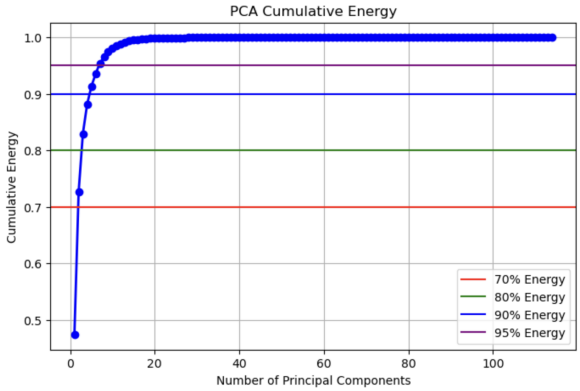
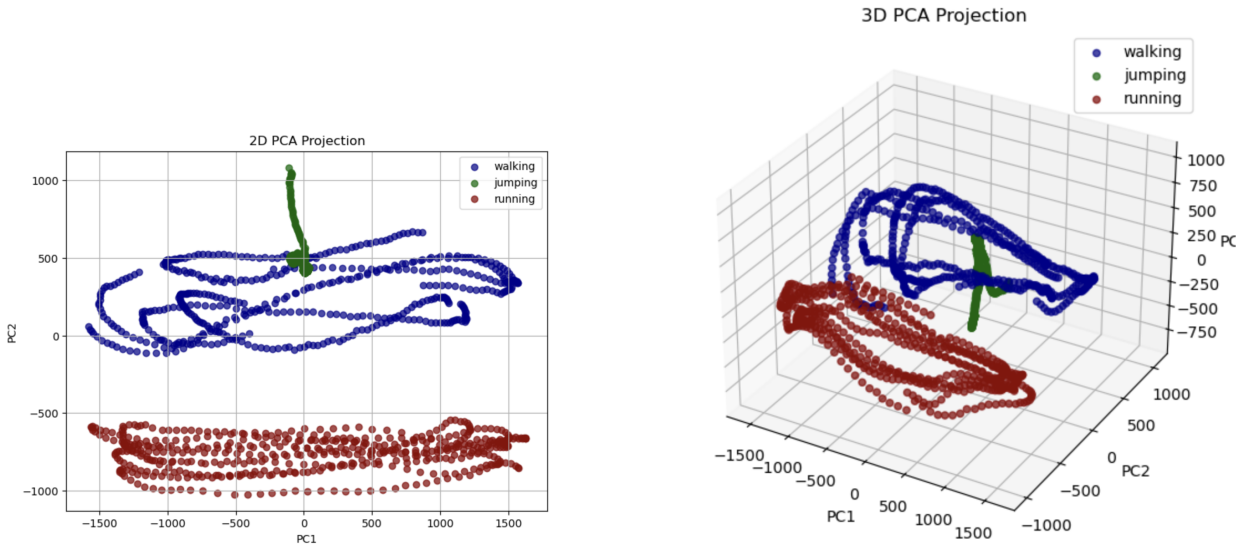


Figure 2. PCA Cumulative Energy

4.2. PCA Visualization and Analysis. To analyze the motion classification in a lower-dimensional space, the dataset was projected onto its first two and three principal components. The 2D and 3D PCA projections, shown in Figure 3a and Figure 3b, effectively separate different movement patterns—walking, jumping, and running—using color-coded trajectories.

The **2D projection** in Figure 3a shows distinct movement clusters: walking (blue) follows a structured path, jumping (green) is vertically dispersed, and running (red) remains compact. The **3D projection** in Figure 3b improves separability, with jumping extending vertically while walking and running stay clustered. These results confirm PCA effectively reduces dimensionality while preserving motion characteristics.



(A) 2D PCA Projection (B) 3D PCA Projection

FIGURE 3. Motion trajectories projected onto 2D and 3D PCA spaces.

4.3. Ground Truth Labels and Centroid Computation. Ground truth labels were assigned: walking (0), jumping (1), and running (2). The labeled data was projected into the k-dimensional PCA space, where centroids were computed. The ground truth labels used for classification are [1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0]. Selecting $k = 2$ retains 80% of the total energy, balancing dimensionality reduction and motion representation. Table 1 shows centroid positions, with walking centralized, jumping displaced along PC2, and running distinctly separated. This confirms PCA preserves essential motion characteristics.

Movement	PC1	PC2
Walking	-36.88	253.35
Jumping	-23.89	499.37
Running	60.77	-752.72

TABLE 1. Centroids in k-modes PCA space for $k = 2$

4.4. Training and Classification Accuracy. Classification accuracy for different k values is shown in Table 2 and Figure 4. Low k values fail to capture motion variability, while higher k introduces complexity, causing accuracy fluctuations. The highest accuracy occurs at $k = 2$, retaining 80% of the energy, balancing dimensionality reduction and classification performance. This choice ensures computational efficiency while maintaining high accuracy, making it a suitable selection for centroid computation in Section 1. Beyond $k = 2$, accuracy fluctuates due to less informative components introducing noise.

4.5. Test Data Classification Performance. Test classification accuracy for different k values is shown in Table 3. The highest accuracy (**98.33%**) occurs at $k = 2$, confirming that two principal components capture essential motion characteristics while minimizing noise. Larger k values introduce fluctuations, likely due to overfitting or redundant features. Figure 5 shows a peak at $k = 2$, with accuracy dropping at intermediate values like $k = 4$ and $k = 6$.

k	Classifier Accuracy (%)
1	50.73
2	88.13
3	75.60
4	73.00
5	75.07
6	72.60
7	87.07

TABLE 2. Classifier accuracy for different values of k

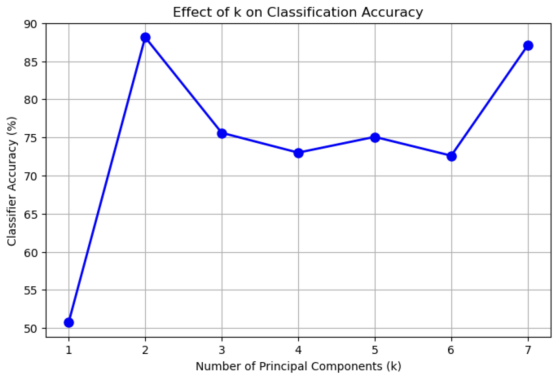


FIGURE 4. Effect of k on classification accuracy

k	Classifier Accuracy (%)
1	48.67
2	98.33
3	92.33
4	74.67
5	91.67
6	71.67
7	94.33

TABLE 3. Test classifier accuracy

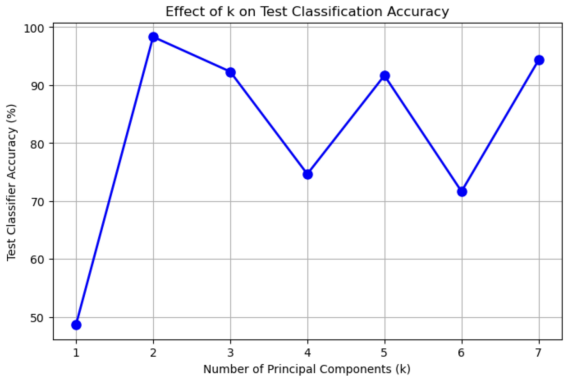


FIGURE 5. Effect of k on test classification accuracy

Kernel	k	Train (%)	Test (%)
RBF	2	35.20	33.33
	3	35.60	33.33
	4	35.93	33.33
	5	36.60	33.33
Polynomial	2	87.13	**98.33**
	3	90.20	92.67
	4	90.80	92.67
	5	94.60	90.33
Sigmoid	2	33.33	33.33
	3	33.33	33.33
	4	33.33	33.33
	5	33.33	33.33

TABLE 4. Comparison of k-PCA Classifier Accuracy

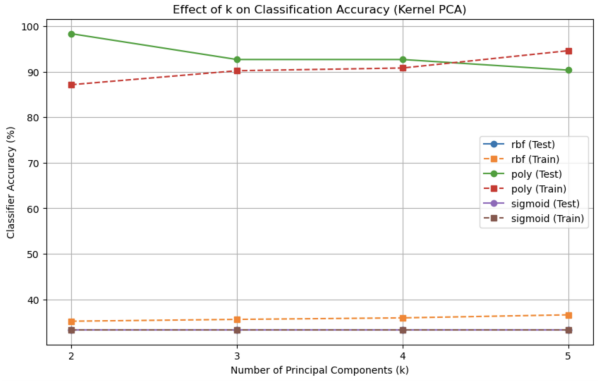


FIGURE 6. Effect of k on Classification Accuracy (Kernel PCA)

4.6. Kernel PCA Classifier Comparison. To assess nonlinear feature transformation, we applied **kernel PCA (k-PCA)** with **RBF, polynomial, and sigmoid** kernels. As shown in Table 4 and Figure 6, the **polynomial kernel** achieves the highest accuracy (**98.33%**) at $k = 2$, aligning

with standard PCA results. In contrast, **RBF and sigmoid kernels** perform poorly, maintaining **33.33%** accuracy across all k, indicating ineffective feature mapping. Thus, the **polynomial kernel at** $k = 2$ is the best alternative, balancing accuracy and feature preservation.

5. Summary and Conclusions

This study applied **PCA** for motion classification in **OptimuS-VD** and and developed a centroid-based classifier to recognize walking, jumping, and running movements. Results showed that $k = 2$ achieved the highest accuracy, preserving essential motion features while minimizing noise. Higher k values led to fluctuations, capturing less relevant variance. The PCA projections confirmed clear separability among movements, validating PCA as an efficient method for real-time classification.

Acknowledgements

This author thanks Professor Frank for providing useful information to tackle the problem through the hints in the assignment, as well as the HW2-Helper.ipynb code. The author is also grateful to TA Rohin Gilman for providing theoretical background to assist in constructing my code. Finally, my peers Liam Lin were also helpful when I ran into issues related to coding.

References

[1] J.N. Kutz, *Methods for Integrating Dynamics of Complex Systems and Big Data*, Oxford, 2013.
[2] N. Frank, *Lecture 11 Notes*, Feb 2 2025.
[3] N. Frank, *Homework 1-TA Slides*, January 2025.