AMATH 482/582: HOME WORK 2

SARAYU GUNDLAPALLI

Applied Mathematics, University of Washington, Seattle, WA
`saru07g@uw.edu`

Abstract. This project develops a motion classification system for a humanoid robot named OptimuS-VD. Motion data consisting of 3D joint coordinates for 3 distinct movements (walking, jumping, running) is collected across multiple samples. Principal component analysis (PCA) is applied to reduce the dimensionality of the high-dimensional motion data and the projections are visualized to examine the separation between movement classes. Centroids for each class are computed in the PCA space and used to classify new samples. Classification accuracy is evaluated on training and test data over varying PCA dimensions. The goal is to enable real-time motion recognition by discerning distinct patterns from high-dimensional sensor data.

1. Introduction and Overview

This report focuses on developing a motion classification system for the humanoid robot OptimuS-VD. The robot is equipped with sensors that record 3D joint coordinate data across 38 joints at a rate of 60Hz. The objective is to leverage this high-dimensional motion data to train a system that can accurately discern between three fundamental movement types performed by OptimuS-VD: walking, jumping, and running.

Using Principal Component Analysis (PCA), we investigate the dimensionality of X_{train}. We determine the number of PCA modes to keep to approximate X_{train} up to 70%, 80%, 90%, and 95% in the Frobenius norm. Centroids for each class are computed in the PCA space and used to classify new samples based on proximity. Classification accuracy is evaluated on training and test data across varying PCA dimensions, with the ultimate aim of identifying the optimal representation for maximizing real-time classification performance.

Motion recognition is a critical aspect of enabling humanoid robots to interact effectively with their environments and human users [5]. The ability to categorize motions from sensor data in real time would allow OptimuS-VD to adapt its control and interactions based on the detected movement type.

2. Theoretical Background

2.1. Singular Value Decomposition (SVD).

A singular value decomposition (SVD) is a factorization of a matrix into several constitutive components. Using SVD, any matrix A (mxm) can be written as the product of a unitary matrix $U \in \mathbb{C}^{m\times m}$, a rectangular diagonal matrix with non-negative diagonal elements $\Sigma \in \mathbb{R}^{n\times m}$ and the conjugate transpose of another unitary matrix $V \in \mathbb{C}^{n\times n}$, V^*. More concretely, the SVD of A is

$$A = U\Sigma V^* \tag{1}$$

Date: February 13, 2024.

The matrices U and V^* are rotational matrices, while Σ is a scaling matrix. There are many theorems regarding the SVD, among the most important of them are the following:

Theorem 1: *For any N so that $0 \leq N \leq r$, we can define the partial sum*

$$A_N = \sum_{j=1}^{N} \sigma_j u_j v_j^* \tag{2}$$

And if $N = min\{m,n\}$, define $\sigma_{N+1} = 0$. Then,

$$\|A - A_N\|_2 = \sigma_{N+1} \tag{3}$$

Likewise, if using Frobenius norm, then

$$\|A - A_N\|_2 = \sqrt{\sigma_{N+1}^2 + \sigma_{N+2}^2 + \cdots + \sigma_r^2} \tag{4}$$

where u_j and v_j are the j-th columns of U and V respectively, σ_j is the j-th diagonal entry of Σ and r = Rank(A).

The SVD gives a type of least square fitting algorithm, allowing us to project the matrix onto low-dimensional representations in a formal, algorithmic way.

2.2. **Principal Component Analysis (PCA).** [3]

PCA is viewed as a dimensionality reduction technique, which means that high dimensional data can be represented using a few coefficients. From a linear algebra point of view, PCA is an application of singular value decomposition (SVD), which produces characteristic features determined by C_X, the covariance matrix of data X. The eigenvectors of C_X are the principal components. Defining C_X if X = $\mathbb{U}\Sigma\mathbb{V}^T$, then we have:

$$C_X = \frac{1}{n-1}XX^T = \frac{1}{n-1}\mathbb{U}\Sigma V^T\mathbb{V}\Sigma^T\mathbb{U}^T = \frac{1}{n-1}\mathbb{U}\Sigma^2\mathbb{U}^T$$

If we project the data onto the left singular values and call the result Y (so $Y = U^*X$) then the covariance matrix of Y is given by:

$$C_Y = \frac{1}{n-1}\Sigma^2$$

where $\Sigma^2 = \Lambda$ where Λ is a diagonal matrix containing the eigenvalues of XX^T.

3. **Algorithm Implementation and Development**

We use the following libraries:

- numpy: For numerical operations and array manipulation.[1]
- matplotlib.pyplot: For data visualization[2]
- sklearn.decomposition.PCA (**scikit-learn**): For Principal Component Analysis (PCA) dimensionality reduction.[4]

Algorithm Implementation/Methodology:

- Load the training data by setting folder path and file names.
- Iterate over 5 samples for each movement type, load the data for each sample, and stack them horizontally to form the X_{train} matrix.
- Fit PCA model to transposed X_{train} matrix and retrieve first 5 PCA modes.
- Compute cumulative energy, determine the number of PCA modes needed for desired percentages of energy
- Task 3:
  - Truncate PCA modes to 2 and 3 modes
  - Project X_{train} onto the truncated PCA spaces
  - Plot in 2D and 3D space

- Task 4:
  - Assign ground truth labels to the samples
  - Perform PCA for k=2 and k=3, compute the mean (centroid) of the samples' projections onto the k-modes PCA space
- Task 5: Define a range of k values for PCA truncation, iterate over each k value
  - Assign trained labels by computing the distance between each sample's projected point and the centroids.
  - Calculate the accuracy of the trained classifier using ground truth labels and trained labels.
- Task 6: Load the test samples, project the test samples onto the same PCA space, and predict the labels based on distances to centroids, compute the accuracy.

4. **Computational Results**

Using PCA to investigate the dimensionality of X_{train}, we observe that the first 5 PCA modes indicate the major patterns of variation or motion in the training data 1a. The PCA modes extract the core motion patterns in the data, the first few dominant modes will reveal the primary ways the joints are moving and coordinating. To approximate X_{train} up to 70%, 80%, 90% and 90% in the Frobenius norm, we keep 2, 3, 5 and 7 PCA modes respectively1. A sharp increase is seen in the first 10 components, indicating they capture the majority of the variance 1b. This suggests the core coordinated movements can be well-approximated with a 10-12 dimensional subspace, there is significant redundancy in the full 114-dimensional coordinate data that can be reduced without much information loss.

After truncation to k-modes, the 2D projection shows partial separation between the movement classes along PC1 and PC2.2a The 3D projection shows a clearer separation between the clusters, with each movement occupying a distinct region of the PC1-PC2-PC3 space.2b Mapping high-dimensional trajectories to low-dimensional PCA space provides an informative motion visualization

No of PCA k-modes	Desired %
2	70%
3	80%
5	90%
7	95%

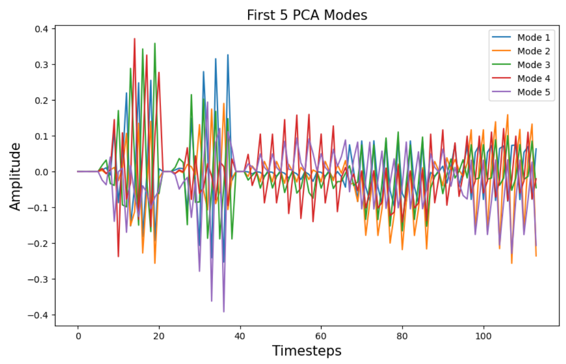
Table 1. Number PCA modes do you need to keep in order to approximate X_{train} up to 70%, 80%, 90% , 95% in the Frobenius norm (i.e., energy)

Centroid(mean) in k-modes PCA space for each movement type is computed.

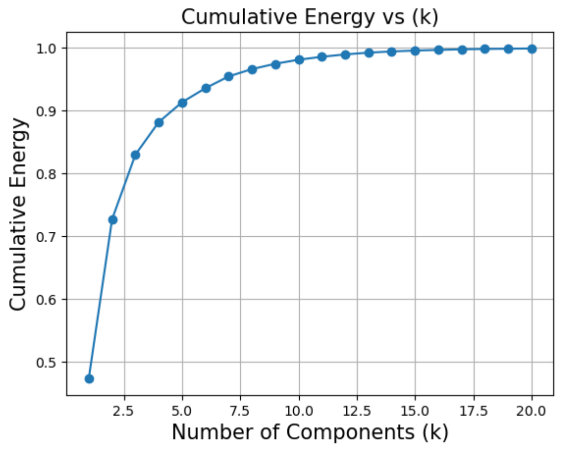
Movement type	Centroids for k = 2, 3
walking	-36.88211143 / 253.35282541 / 175.91202104
running	60.77197779 / -752.7210869 / -103.41194553
jumping	-23.88986635 / 499.36826149 / -72.5000755

Table 2. centroid(mean) in k-modes PCA space for each movement type

Further, we compute the trained labels for various k values of k-PCA truncation and look at the accuracy of the trained classifier. Accuracy increases rapidly as the number of PCA dimensions used increases from 2 to around 10, this aligns with earlier analysis showing intrinsic dimensionality around 10-121b. This justifies the tradeoff between using enough dimensions to sufficiently encode motion patterns versus too many that lead to overfitting. Accuracy peaks at around 91% with 14



(A) First 5 PCA modes



(B) Cumulative energy vs k

FIGURE 1. Task1, Task2

PCA dimensions.33 So, 14 dimensions encode the core aspects needed for good classification, this would also be the optimal k for classifier accuracy.

No of PCA k-modes	Trained accuracy	Test accuracy
2	0.8813	0.983
3	0.756	0.9233
5	0.7506	0.9166
7	0.8706	0.9433
9	0.8786	0.9433
10	0.888	0.9433
12	0.9093	0.9533
14	0.9106	0.9533

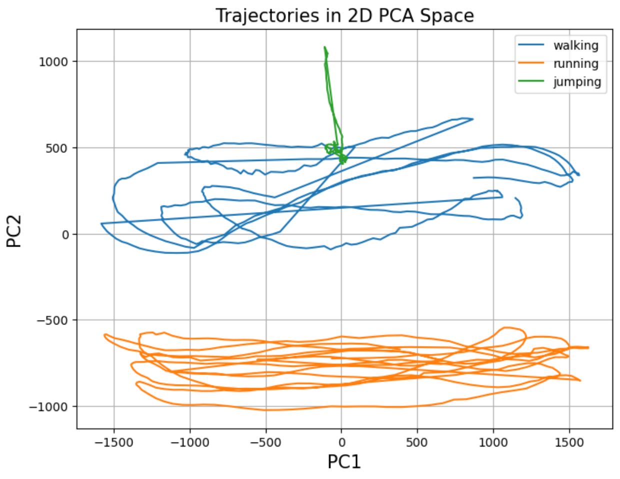
TABLE 3. Comparison of the trained accuracy and test accuracy w. the classifier

Looking at the accuracy of the classifier on the test samples, we see that test accuracy peaks at around 95% with 12 PCA dimensions, indicating that the test data converges faster, and is more accurate. This is a clean and small dataset, so it is possible for the test accuracy to be higher. 33

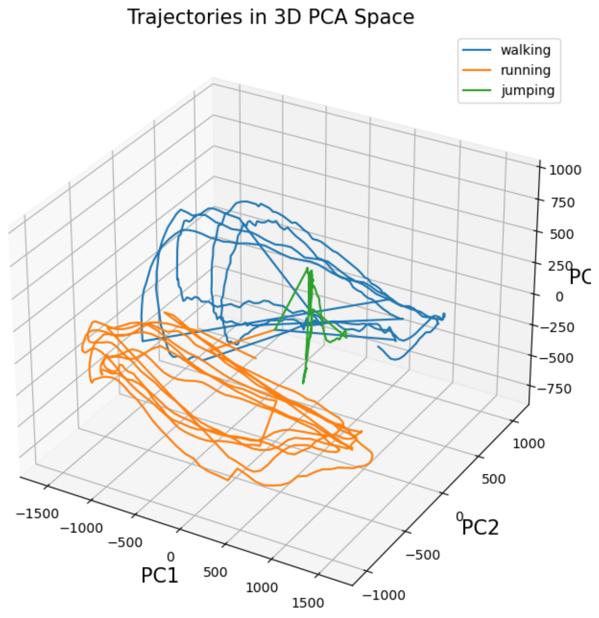
5. Summary and Conclusions

PCA was applied to human motion capture data consisting of 3 movements (walk, run, jump) with 5 samples each. Visualizing the first 5 PCA modes showed they capture major patterns of coordination and periodicity, mapping high-dimensional motion trajectories into low-dimensional PCA space enables insightful visualization of the major modes of variation. The cumulative energy plot revealed a sharp increase in the first 10-12 components, followed by a plateau. Projecting onto 2 PCA dimensions showed partial class separation, while 3 dimensions better disambiguated the movements. Trajectories in PCA space followed cyclic paths reflecting the periodic nature of the motions. Around 14 modes were needed to capture 91% accuracy on the trained data while test accuracy exceeded training accuracy peaking at 95% (k=12), suggesting a very clean and possibly insufficient dataset.

Overall, PCA offers a pipeline for dimensionality reduction, motion visualization, and data-driven learning on motion capture datasets. Further experiments with larger, balanced datasets would



(A) Trajectories of each movement type in 2D PCA space



(B) Trajectories of each movement type in 3D PCA space

FIGURE 2. Task3

provide deeper insight into generalization performance and enable real-time motion recognition by discerning distinct patterns from high-dimensional sensor data.

Acknowledgements

The author is thankful for the suggestions/advice about the algorithm implementation from peers in AMATH 482/582. The author is thankful to the Teaching Assistants, Saba Heravi and Juan Felipe Osorio Ramírez, for their invaluable time and fruitful discussions about setting up the classifier. Further, the author is thankful to Prof. Eli Shlizerman for useful discussions about the course content.

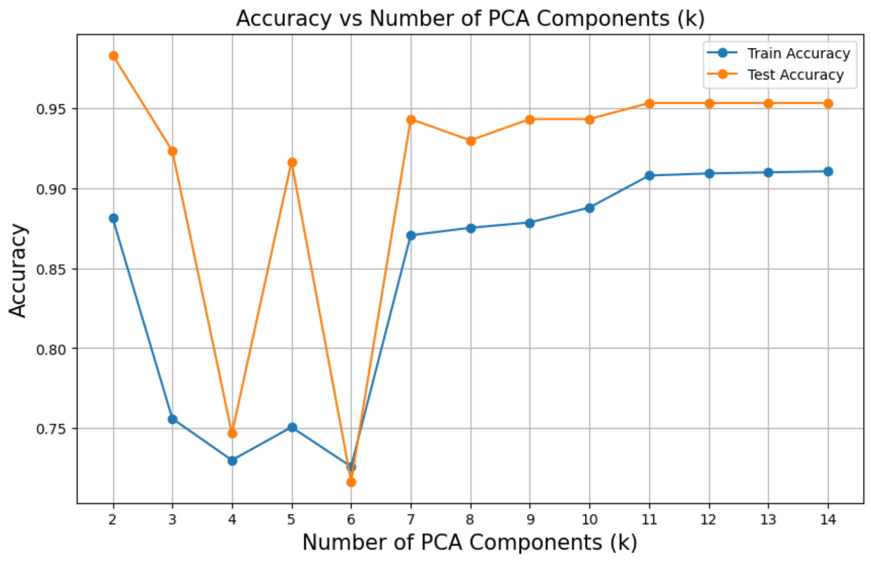


FIGURE 3. Accuracy of the classifier for trained data and test data vs the number of PCA modes (k)

References

[1] 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.

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

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

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[5] G. N. Tripathi and H. Wagatsuma. Pca-based algorithms to find synergies for humanoid robot motion behavior. *International Journal of Humanoid Robotics*, 13(02):1550037, 2016.