# AMATH 582: HOMEWORK 2

HUNTER LYBBERT

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

Abstract. In this analysis of the joint movement data of OptimuS-VD we experiment with Principal Component Analysis (PCA) as a means of dimensionality reduction. Our goal was to build a projection of the recordings to a lower dimension than the number of coordinates, visualize the movements in this new lower dimension. Using the lower dimensional projection we design an algorithm that recognizes which movement OptimuS-VD is performing. Furthermore, we first implemented a clustering-esque classifier and evaluate it's accuracy using the training set as well as the held out test set. Considerations of further work are given.

## 1. Introduction and Overview

There are many computationally intensive computer vision models (primarily using convolutional neural-networks) that perform well on object and movement recognition. This was all kickstarted by a paper published in *Advances in Neural Information Processing Systems* by Krizhevsky et. al. [3]. In our analysis we do not make use of neural networks or deep learning, we will primarily use the Singular Value Decomposition (SVD) via PCA. In order to understand the data and attempt to classify the movements of the robot we aimed to complete the following 5 tasks:

(1) Perform PCA on our robot movement data such that PCA modes are spatial modes and coefficients are time-dependent coefficients. Investigate how many PCA spatial modes you need to keep to approximate $X_{\text{Train}}$ up to 70%, 80% , 90% , 95% in Frobenius norm (i.e., energy) and plot results.

(2) Truncate the PCA modes set to 2 and 3 modes and plot the projected $X_{\text{Train}}$ in the truncated PCA space as low dimensional 2D (PC1,PC2 coordinates) and 3D (PC1,PC2,PC3 coordinates) trajectories discuss visualization and your findings.

(3) In order to classify each sample with type of movement establish a ground truth label for each frame from each movement sample. Then for each movement compute its centroid (mean) in $k$-modes PCA space.

(4) Having the ground truth, create a classifier which predicts labels each sample based on the which movement type centroid it is closest to. Compute these predicted labels for various $k$ values of $k$-PCA truncation and report the accuracy of the trained classifier (the percentage of samples for which the ground truth and the trained labels match). Discuss your results in terms of optimal $k$ for the classifier accuracy.

*Date*: February 11, 2025.

(5) Apply this classifier to the test samples. Report the accuracy of the classifier on the test samples. Discuss and compare it with trained accuracy. Try various $k$ values.

(6) Bonus (+2 points): Implement an alternative classifier based on $k$-PCA space and compare with your results above.

In the endeavor to reduce the dimensionality of the robot movement data, classify the movements, and complete the requisite tasks, we made extensive use of several important Python packages. Namely, Matplotlib was used to create all plots and animations [2]. Additionally, Scikit-learn was the primary source of using the PCA algorithm and other classification methods [4]. Finally, NumPy was once again a crucial tool [1]. Moreover there are important theoretical underpinnings behind the algorithm we implemented which will be cited and expounded upon in the following section.

## 2. Theoretical Background

Technical background duh duh duh ... **TODO**

$$X = U\Sigma V^T \tag{1}$$

Let's get into the actual implementation now.

## 3. Algorithm Implementation and Development

We will now describe in words and pseudocode the implementation of these methods as we used them in this application to reduce the dimensions of our data and try to preserve as much information as possible. First, we discuss the PCA algorithm in 1.

**Algorithm 1** Determine the Dominant Frequency

```
$S = \text{np.load(data)}$    ▷ Input subdata, after reshaping to (64, 64, 64, 49)
$\hat{S} = \text{fftn(S)}$
$\hat{S}^\dagger = \text{fftshift}(\hat{S})$    ▷ Transformed and shifted subdata (64,64,64,49)
$\hat{S}^\dagger_{\text{avg}} = \text{avg}(\hat{S}^\dagger)$    ▷ Average computed across time at each point (64,64,64)
$x, y, z = \text{argmax}\Big(\text{abs}\big(\hat{S}^\dagger_{\text{avg}}\big)\Big)$
```

The result of the final step of Algorithm 1 are **TODO: fill in deets**. What we really want is to know what the frequency value should be to center our gaussian filter from equation (1) to use in Algorithm 2

**Algorithm 2** Apply Gaussian Filter in Frequency Space

```
$G = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\Big(-\frac{1}{2\sigma^2}\big(k_x - k_{x_0}\big) + \big(k_y - k_{y_0}\big) + \big(k_z - k_{z_0}\big)\Big)$    ▷ Gaussian Filter using (1)
$\hat{S} = \text{fftn(S)}$
$\hat{S}^\dagger = \text{fftshift}(\hat{S})$
$\hat{F}^\dagger = \hat{S}^\dagger G$    ▷ Filtered subdata in frequency space still
$\hat{F} = \text{ifftshift}(\hat{F}^\dagger)$
$F = \text{ifftn}(\hat{F})$    ▷ Filtered subdata in signal space now
```

**TODO:**. These results will be described further in following section.

## 4. Computational Results

We have the following to talk about:

- Talk about getting the right shape for training
- That means we had to treat each frame of the robot movements as a single sample
- Talk about the classification issues
- why we used the Support vector machine to try and classify more accurately

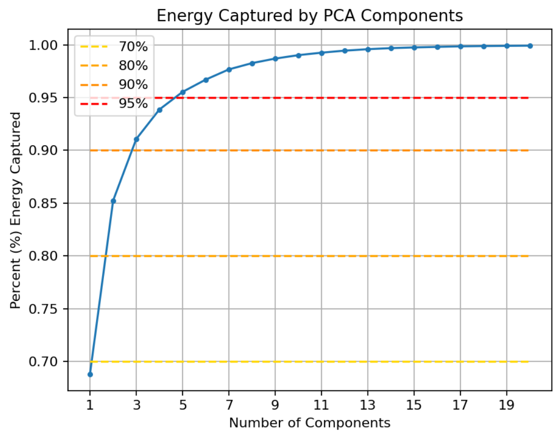
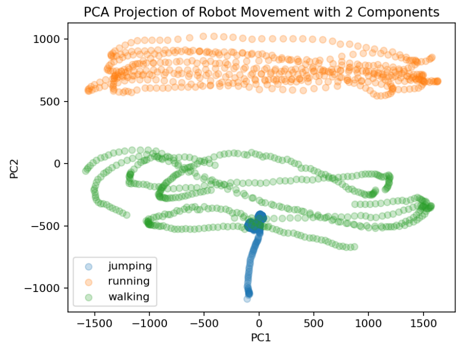
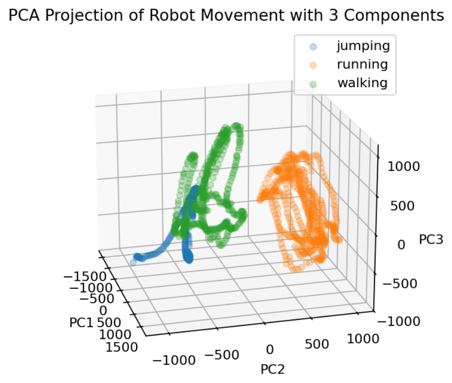


Figure 1. Visualizing each of the 3 slices including the location in our 3 dimensional average frequency object. We also visualize a slice which does not intersect the max frequency in order to convey how drastic the location of the max frequency is. In order to show this comparison things have been rescaled here.



(A) First image caption



(B) Second image caption

Figure 2. Overall figure caption describing both images

As seen in Figures 1 and 2, then Figure 3.

## 5. Summary and Conclusions

This is my summary and conc. **TODO:**

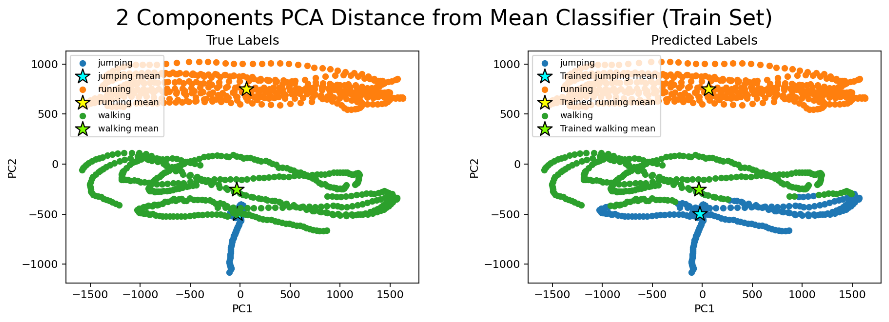


FIGURE 3. The resulting path in 3 dimensions that we determined after applying Algorithms ?? and ??. In this iteration of the filter we used $\sigma = 1.3$.

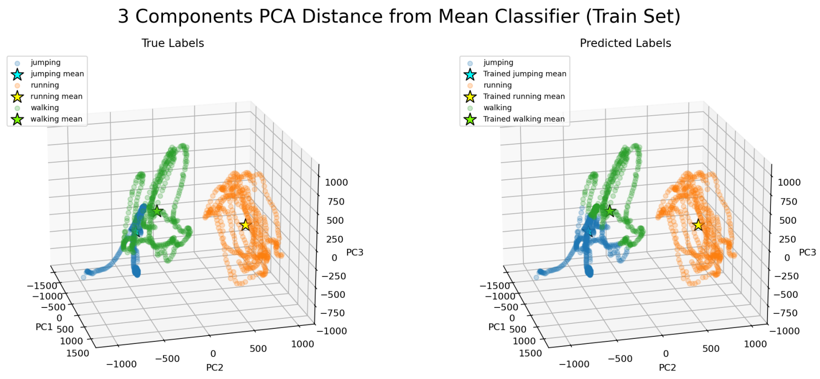


FIGURE 4. This is just the 2 dimensional projection of the path given in the above 3d plot. The same value of $\sigma$ was used in the filter.

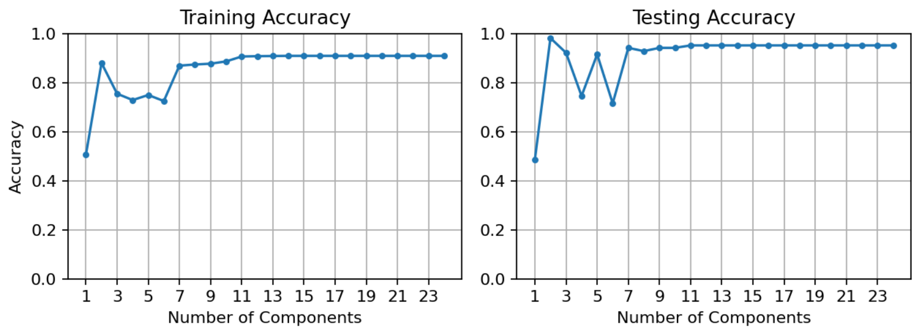


FIGURE 5. Visualizing each of the 3 slices including the location in our 3 dimensional average frequency object. We also visualize a slice which does not intersect the max frequency in order to convey how drastic the location of the max frequency is. In order to show this comparison things have been rescaled here.

## ACKNOWLEDGEMENTS

The author is thankful to Jaxon Tuggle for useful discussions about the process to find the correct dimensions to use when interpreting the training data samples we were given. We would also like to thank Professor Eli Shlizerman for carefully instructing us in class. Finally, it is necessary to thank

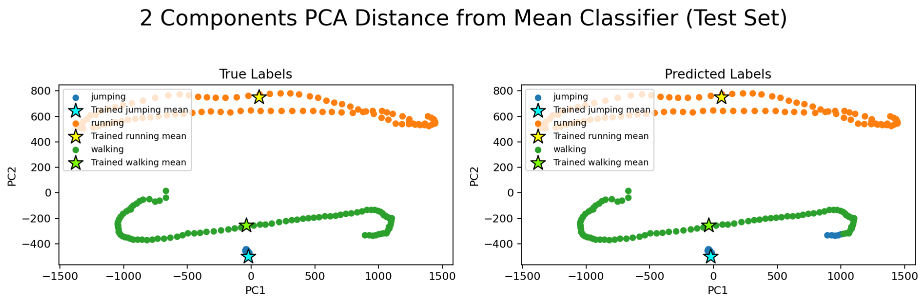


FIGURE 6. Visualizing each of the 3 slices including the location in our 3 dimensional average frequency object. We also visualize a slice which does not intersect the max frequency in order to convey how drastic the location of the max frequency is. In order to show this comparison things have been rescaled here.

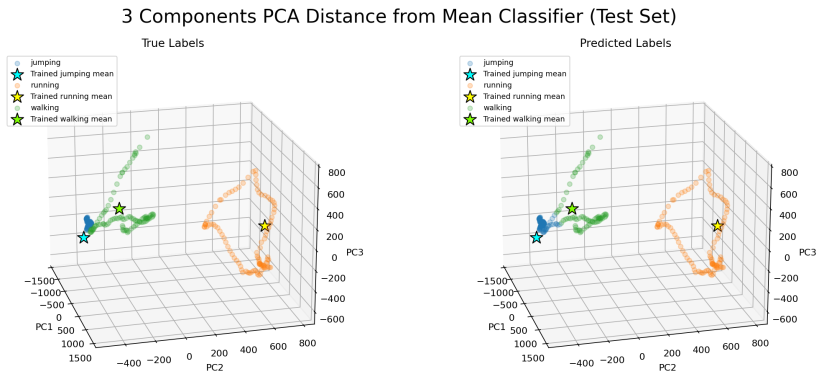


FIGURE 7. Visualizing each of the 3 slices including the location in our 3 dimensional average frequency object. We also visualize a slice which does not intersect the max frequency in order to convey how drastic the location of the max frequency is. In order to show this comparison things have been rescaled here.

the following students Nate Ward, Sophie Kamien, whose questions helped clarify understanding of the algorithm we implemented by giving chances to explain ideas and debug code together.

## 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] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.
[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.