HW2: CAPTURED MOTION RECOGNITION

TIANBO ZHANG

University of Washington, Department of Applied Mathematics, Seattle

Abstract. This report details the process of capturing motion data, applying PCA for dimensionality reduction, and designing a classifier for real-time movement recognition. We will test our classifier and show the predication performance at the end.

1. Introduction and Overview

Introduction: In the realm of robotics, the development of humanoid robots like OptimuS-VD represents a fascinating leap towards creating machines that mirror human capabilities and interactions. However the problem of motion recognition has become a huge challenge in this area. Hence, we will design a classifier for real-time movement predication to enhance the functionality of our robot.

Overview: Our data set is given as Euler angles that can be transformed to xyz coordinate. We have 5 recorded samples of each of the 3 movements (walking, jumping, running). Then we will perform the following steps:

(1) Compile X_{train} and use PCA to investigate the dimensionality of X_{train}.
(2) Investigate how many PCA modes do you need to keep in order to approximate X_{train} up to $70\%, 80\%, 90\%, 95\%$ in the Frobenuis norm.
(3) Truncate the PCA modes to 2 and 3 modes.
(4) Create ground truth labels with: 0(walking), 1(jumping), 2(running). Assign the labels and compute the centroid.
(5) Train our classifier.
(6) Test the classifier by predicting the test labels and compare with trained accuracy.
(7) Implement an alternative classifier based on k-PCA.

2. Theoretical Background

2.1. Frobenius norm.

Frobenius norm is a matrix norm (not a induced norm) be defined as following:

$$||A||_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}|a_{ij}|^2} = \sqrt{Tr(A^TA)} = \sqrt{Tr(AA^T)}$$

2.2. Singular Value Decomposition.

For any $A \in \mathbb{R}^{n*m}$ there exists unitary matrices $U \in \mathbb{R}^{n*n}, V \in \mathbb{R}^{m*m}$ and a diagonal matrix $\Sigma \in \mathbb{R}^{n*m}$ with positive entries such that:

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

Note we will have the following with sequence of transformations with SVD:

$$Ax = U\Sigma V^T x \Rightarrow x \xRightarrow[\text{rotation}]{V^T} V^T x \xRightarrow[\text{scaling}]{\Sigma} \Sigma V^T x \xRightarrow[\text{change of basis}]{U} U\Sigma V^T x$$

2.3. **Principle Component Analysis.** PCA aims to measure the variance of the data in each axis that is centralized(mean 0). It can also find a decent solution to the "least number of components to summarize data". We can start by considering the covariance matrix:

$$C_x = Cov(x) \cong \tfrac{1}{N-1} XX^T$$

Then plug in the SVD of X we have:

$$C_x \cong \tfrac{1}{N-1} U\Sigma V^T V \Sigma U^T = \tfrac{1}{N-1} U\Sigma^2 U^T$$

Then let $Y = UX$, we have:

$$C_Y = \tfrac{1}{N-1} YY^T = \tfrac{1}{n-1}\Sigma^2$$

Then note PC modes are eigenvectors of C_x and σ^2 are eigenvalues of C_x. There fore, PC modes indicate the "optimal" directions to represent variance of the data.

3. Algorithm Implementation and Development

3.1. **SVD computation.** After loading the data from our sample files, we can proceed to calculate the SVD for PCA analysis. Before any computation we have to centralize our data as following:

$$X_{centered} = X_{\text{ith row}} - \overline{X}_{\text{ith row}}$$

Then we can compute the singular value decomposition using function from numpy.

3.2. **PCA modes investigation.** To find the appropriate amount of PCA modes for different percentage of approximation, we have to implement the Frobenius norm. In other words we have to find the cumulative energy. Note we can compute the energy for each PCA modes using following formula:

$$E = \tfrac{\sigma^2}{\sum \sigma^2}$$

Then we have the following algorithm:

Algorithm 1 Energy Calculation and PCA Modes Determination

```
Compute the energy for each row
Set the cumulative energy to zero
for each PCA component's energy do
    Compute the cumulative energy up to this PCA mode
    if cumulative energy is bigger than the targeted percentage then
        Return the index of the PCA mode
    end if
end for
```

3.3. **PCA projection.** U effectively provides a basis onto which the data can be projected. Hence to project our data onto k-PCA space, we can perform the following step:

$$X_{approx} = U_{[1:k]}^T X$$

3.4. **Create ground truth labels.** We can create the labels for each movement as following: 0 for walking, 1 for running, and 2 for jumping. Then we can assign these labels to each sample in our data set.

3.5. **Train classifier.** To train our classifier we need to compute the centroid for each movement for the first k PCA modes. We can compute the centroid by taking the mean. Than we can assign the data with the label that has the least distance from the centroid in the PCA projection.

Algorithm 2 Train Classifier

```
Set up the empty predicted label list
for the every k values we want to try do
    Project our dataset onto k-PCA space
    Compute the centroids for each movements for k PCA modes
    for each sample in the data set do
        Find the movements that has the closest distance from the
centroid to the sample
        Append the predicated label to the label list
    end for
    Compute the accuracy using function from sklearn
end for
```

3.6. **Perform label predication on test dataset.** After fetching the data from the sample files, we need to find the PCA projection for the data. Then we can use the same SVD and movement centroids from train dataset and predict the labels by finding the least distance between samples and movement centroids.

4. Computational Results

4.1. **PCA modes in xyz plane.** After compile all the train samples into a matrix X_{train}, we can compute the PCA modes by taking the U unitary matrix from X_{train}'s SVD. Then we can check the first PCA modes by graphing them as following:

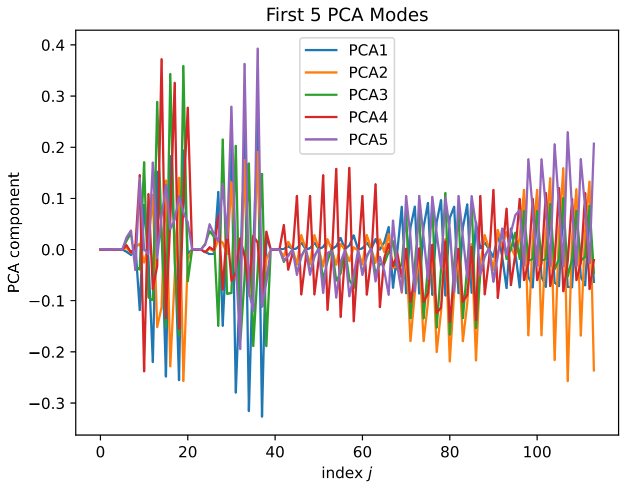

FIGURE 1. First 5 PCA modes as vector vs. indeces

Observe that from the graph we can see that every PCA mode shows a similar patterns.

4.2. Cumulative Energy.

Then after the computation of the PCA modes we can find the amount of PCA modes we needed in order to keep the approximate X_{train} up to a certain percentage by computing the energy for each mode and cumulatively add them up. Implementing algorithm 1, we will have the following graph.

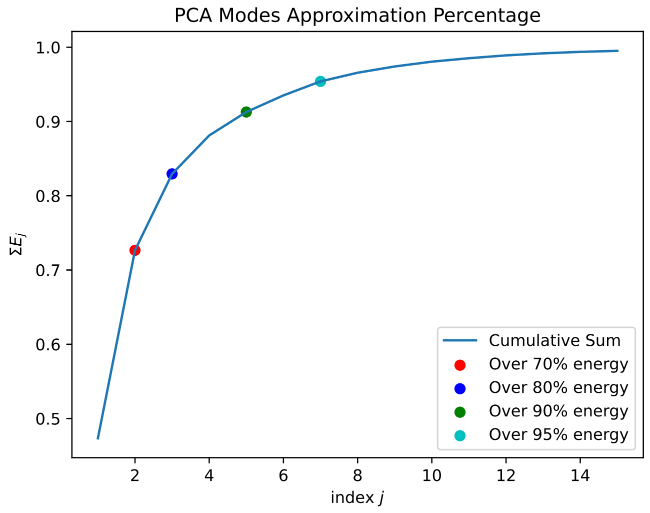

FIGURE 2. Cumulative energy for PCA modes

As we can see from the graph if we want to approximate X_{train} up to 70% we need to use 2 PCA modes, 80% we need to use 3 PCA modes, 90% we need to use 5 PCA modes, 95% we need to use 7 PCA modes.

4.3. Data projection on 2 and 3 PCA space.

After implementing 3.3 to project our X_train onto k-PCA space we can graph the projected movements in 2 PCA and 3 PCA space.

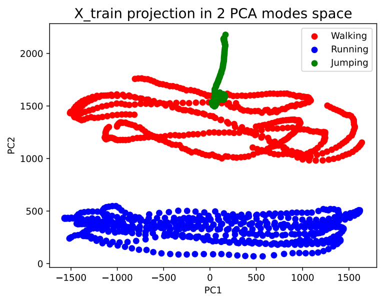

FIGURE 3. 2PCA

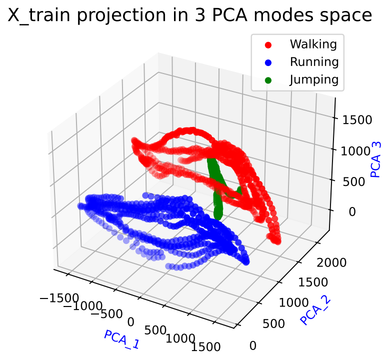

FIGURE 4. 3PCA

4.4. **Classifier Application.** Lastly we can train our label classifier by implementing Algorithm 3. Then we can compare with the test accuracy with train accuracy as following:

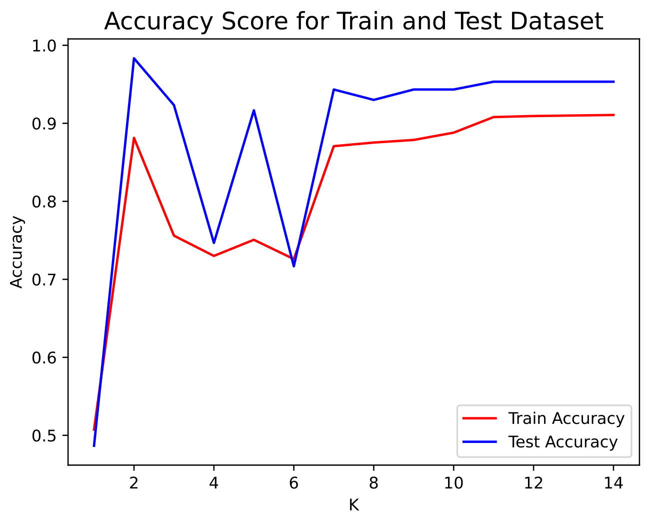

FIGURE 5. Accuracy Score for Test and Train

As we can see in the figure that the accuracy score for test and train datasets shows a similar pattern. Both score goes up and down until after 7 PCA modes. However for 3 PCA modes and 4 PCA modes classifier, the result has a different turning point where test dataset has a steeper decrease on accuracy after 3 PCA modes, but the train dataset has a smoother decrease after 3 PCA modes.

Finally, we will choose 7PCA modes to perform classification, since it has a accuracy of 94.33 percent for test dataset and 87.1 percent for the train dataset. And it has a simpler structure compare to higher degree of PCA modes which only have 2 or 3 percent higher accuracy.

5. Summary and Conclusions

In conclusion the implementation of Principle Component Analysis for dimensional reduction and classifier design offers significant utility for real-time movement recognition.

In addition, we found the optimal PCA modes (7 PCA modes) to effectively predict the labeling of our dataset. This choice is justified over higher degrees of PCA modes that offer only marginally higher accuracy, thus underscoring the efficacy of PCA in movement classification tasks within the context of enhancing humanoid robot functionalities.

Lastly, we implement the Random Forest Classifier for comparison. Although we resulted in a much higher accuracy of 100 percent, the problem of over fitting our data is another problem we need to consider in the future.

Hence the approach of PCA projection successfully reduced the dataset to its most significant components, enabling the classification algorithms to perform efficiently. This methodology not only demonstrated high accuracy in classifying movements but also offered insights into the potential for further research in movement recognition and its applications in various fields such as robotics, physical therapy, and animation.

6. **Acknowledgements**

The author is thankful to Pro.Zhang from Beijing University of Post and Telecommunications for useful discussions about potential methods that are based on Principle Component Analysis

7. **References**

References

[1] J.N. Kutz (2013) *Methods for Integrating Dynamics of Complex Systems and Big Data*, Oxford.
[2] Alan V. Oppenheim, Alan S. Willsky (2022) *Signals and Systems*, 2nd Edition.
[3] Anshul Saini (2022) *An Introduction to Random Forest Algorithm for beginners*, Analytic Vidhya.

8. **Appendix (Extra Credit)**

We will implement the Random Forest Classifier as our alternative classifier that's based on k-PCA space. From upper research we found that 7 PCA modes is well enough to capture the patterns in our dataset. Hence, we will implement the following algorithm:

Algorithm 3 Random Forest Classifier Application

```
Found the first 7 PCA modes from our train dataset
Project $X_{train}$ and $X_{test}$ onto 7 PCA modes space
Train our Random Forest model using projected $X_{train}$
Predict our labels for $X_{train}$ and $X_{test}$
Compute the accuracy score for train and test dataset
```

Note we have a accuracy score of 100 percent for both the train and test dataset. Compare to our classifier, the accuracy is indeed much higher. Although the result seems tempting, we still need to take the account for over fitting. And also there should be further discussion around the hyper-parameter of n_components.