AMATH 582: Homework 2

Kutay Demiralay

Aeronautics and Astronautics Department Department, University of Washington, Seattle, WA

In this homework assignment, a classifier is developed successfully to discern between three distinct movements performed by a humanoid robot: jumping, running, and walking. Singular Value Decomposition (SVD) is applied to project the recorded movements into a lower-dimensional space. The principal component analysis (PCA) modes of the training data are examined and visually represented. Different values of k in k-PCA truncation are utilized to project the training data into truncated PCA spaces. The accuracy of the designed classifier is computed for both training and testing datasets across various k values of k-PCA truncation, providing insights into the effectiveness of the classifier under different dimensionality reductions.

Introduction and Overview

Each collection of training data encompasses details from 38 joints of a humanoid robot, capturing positional information in the x, y, and z coordinates over a sequence of 100 time steps. The observations are recorded at intervals of 1.4 seconds, leading to the formation of a 114x100 matrix for each set. Altogether, there are 15 unique sets of Training data under examination, each comprising 5 samples related to the robot's performance of three distinct movements: walking, jumping, and running. Another 3 different sets of Test data are available for us to test the classifier we designed.

Theoretical Background

Principal Component Analysis (PCA) is a mathematical technique employed for reducing the dimensionality of data sets while preserving essential patterns and variations. It operates by identifying the principal components within the dataset, which are the directions where the data shows the most significant variability. The data is linearly transformed onto a new coordinate system.PCA helps identify the most influential aspects, sort of like finding the most critical features or trends that capture the essence of the data. Basis for PCA can be obtained using Singular Value Decomposition shown in eq.(1), which is a tool that helps us to decompose the data matrix into components that reveal the inherent structure and variability. Where the U matrix contains the left singular vectors, Σ is a diagonal matrix of singular values and V^T contains the right singular vectors.Singular values are the positive square roots of the eigenvalues of the covariance matrix of A or the matrix multiplied by the transpose of matrix SVD is a sequence of 3 transformations as can be seen in eq. (2), rotation, scaling, change of basis Principle Components.

$$X = U\Sigma V^T \quad (1)$$

$$Ax = U\Sigma V^T : x \xrightarrow[\text{rotation}]{V^T} V^T x \xrightarrow[\text{scaling}]{\Sigma} U\Sigma x \xrightarrow[\substack{\text{change of basis} \\ \text{Principal Components (PCA)}}]{U} U\Sigma V^T x \quad (2)$$

Principal Component Analysis can be obtained in 3 steps.:

-First, we have to center the Data by subtracting the mean of each feature (column) from the corresponding values in the dataset.
-Secondly, apply SVD on centered data to acquire U ,Σ , V^T matrices.
-Thirdly, Principal components are the columns of matrix U, you select the first n columns of matrix U to get first n principle components

After Applying PCA, now that we have the truncated U matrix with top k desired columns being non-zero only. If you multiply the transpose of this U matrix with X data you get data values projected into k-PCA truncated space, where k depends on the user's choice.The classification algorithm works by computing the distance between projected points in k-modes PCA space and each of the centroids. Centroids are mean ink-modes PCA space. The minimum distance determines to which class the sample belongs.

$$X_{projected_k} = U^T{}_{k-Truncated}\, X \quad (3)$$

The Energy can be computer by the square of Frobenius Norm of the matrix, which is also equal to multiplication of all the singular values square, as it can be seen in eq.(4)

$$E = ||A||_F^2 = \sum_{j=1}^{\min(m,n)} \sigma_j^2 \quad (4)$$

Algorithm Implementation and Development

All coding was done in the python programming language. Numpy, Matplotlib, Scipy, Mpl_toolkits, Math, and Sklearn libraries were used.

First Step: Compiling all the train and test samples into matrixes X_train and X_test

Creating a single 114x1500 Matrix from by adding 15 different training data set, each one with 114x100 dimension back to back using for loop, to create single X_train matrix, same is done with Testing data to create a 114x300 shaped matrix from 3 different test data sets shaped 114x100.

```
fname=["walking_2","walking_3","walking_4","walking_5","jumping_1","jumping_2","jumping_3","jumping_4","jumping_5", "running_1","running_2","running_3","running_4","running_5" ]
# Define the folder where the motion data is stored
folder = '/content/drive/MyDrive/train/'
X_train=np.load((folder+"walking_1"+".npy")) #initializing compile process
for name in fname:
  value= np.load((folder+name+".npy"))
  X_train= np.hstack((X_train,value))
```

Second Step: Singular Value Decomposition

First we reshaped the X_train data by centering the data around its mean. Then using phytons built in SVD command, U ,Σ , V^T matrices are found with given the names dU, ds1, and dVt.

```
X_reshaped=X_train
centered_data = X_reshaped - np.mean(X_reshaped, axis=1)[:, None]
dU, ds1, dVt = np.linalg.svd(centered_data)
```

Third Step: Plotting the first 5 PCA modes in xyz space

Plotting Principal components which are basically the columns of matrix U obtained from SVD.

```
for i in range(5):
```

```
 ax.plot(dU3[:, i], label=f' PCA Mode:{i+1}')
```

Fourth Step: Plotting Cumulative Energy

Energy of the matrix is found with respect to PCA modes in Frobenius mode. Cumulative energy vs. mode number is plotted by using np.cumsum command.

```
E = np.power(ds1,2)/np.sum(np.power(ds1,2)) #Energy
plt.plot(np.cumsum(E)[:20]) #Cumulative Energy
```

Fifth Step: Projecting Training Data into Truncated PCA space

Truncated the PCA modes into 2 and 3 modes, projected data to truncated PCA space and plotted the projected X_train in truncated PCA space. Used different plots for each different movement type.

```
X_Projection = np.dot(dU2.T, X_reshaped1) #Truncated the PCA modes to 2 modes
```

Sixth Step: Training Classification Algorithm

Created a vector of ground truth labels with an integer per class, e.g., 0 (walking), 1 (jumping), 2 (running) and assign an appropriate label to each sample in training data.Then for each movement centroid (mean) in *k*-modes PCA space are computed. Distance between the projected point in *k*-modes PCA space and each of the centroids are computed for every k values in k-PCA truncation. The minimal distance will determine to which class the sample belongs to. Accuracy of classification algorithm is computed using a scoring system where the system gains a point for every correct classification and plotted with respect to k values of k-PCA truncation.

```
for k in range (20):
 accuracyscore[k]=0
 centered_Xtrain = X_train-np.mean (X_train, axis=1) [:, None]
 dU, ds, dVt = np.linalg.svd (centered_Xtrain)
 dU[:,k+1:None]=0
 projecteddata=np.dot (np.transpose(dU4) , centered_Xtrain)
 centroid[:,0]=np.mean (projecteddata [:, 0:500], axis=1)
 centroid[:,1]=np.mean (projecteddata [:,500:1000], axis=1)
 centroid[:,2]=np.mean (projecteddata[:, 1000:1500], axis=1)
for i in range (1500):
  distance0=distance. euclidean(projecteddata[:, i], centroid [: ,0])
  distance1=distance.euclidean(projecteddata[:, i], centroid [:, 1])
  distance2=distance. euclidean (projecteddata[:, i], centroid [: ,2])
  index[i]=np. argmin([distance0,distance1,distance2])
  if index[i]==0:
    trainedlabel[i]=0
  elif index[i] ==1 :
    trainedlabel[i]=1
  elif index[i] ==2 :
```

```
    trainedlabel[i]=2
#computing the accuracy of the trained classifier
 for i in range (1500):
   if trainedlabel[i]==groundtruthlabel [i]:
     accuracyscore[k]=accuracyscore[k]+1
```

Sixth Step: Testing Classification Algorithm

The same procedure as the sixth procedure is done with Testing data to test the algorithm and plot the results. The difference with Testing is that we centered the data set around the mean training data set, and we used the U matrix coming from SVD of Training data set, when projecting the test data into PCA space.

Computational Results

First 5 PCA modes in xyz space, generated from the Training data set, can be seen in fig. (1) below.

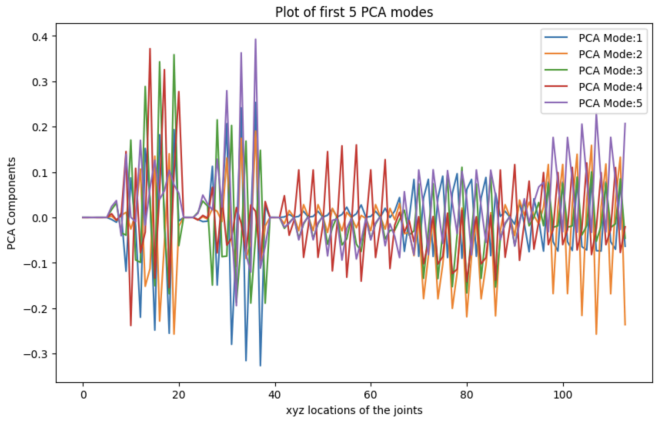

Figure 1: Plot of first 5 PCA modes in xyz space.

To figure out how many PCA modes need to be used in order to approximate Training data into desired level of percentage in energy, Cumulative vs. Number of PCA modes have been plotted. Table 1 shows the plot in figure 2 in table format. The observation reveals that the initial PCA modes bear the highest significance in capturing data, and as the number of PCA modes increases, their relevance in representing the data diminishes exponentially. To capture %95 in the Frobenius Norm only first 6 modes out of the total 114 modes are enough. With only 6 PCA modes we can approximate the training up to %95 in Frobenius norm.

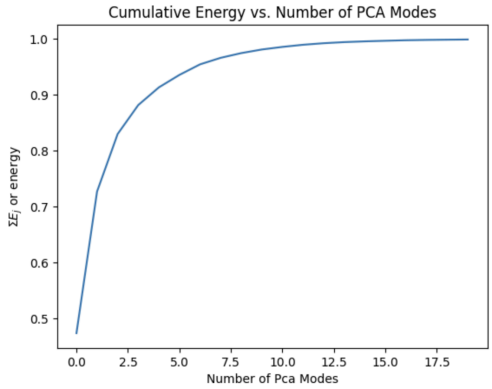

Figure 2: Cumulative Energy vs. Number of PCA Modes

Energy	Number of PCA Modes
0.7265	1
0.8295	2
0.8812	3
0.9128	4
0.9352	5
0.9540	6

Table 1: Cumulative Energy vs. Number of PCA Modes

Truncated the PCA modes to 2 and 3 modes and plot the projected *Xtrain* in truncated PCA space as low dimensional 2D (PC1,PC2 coordinates) and 3D (PC1,PC2,PC3 coordinates) trajectories. Used colors for different movements. Figure 4 is the plot for 2D case and Figure 5 is the plot for 3D case. It is seen that both in 3D dimensional 2D dimensional cases, walking, jumping, running movements have generally different PC coordinates. In the 2D case, the running movement exhibits the lowest PC2 coordinates, with walking occupying the middle ground and jumping having the highest PC2 coordinates overall. The plot for running does not overlap with those for jumping and walking. However, there is a minimal overlap between the jumping and running plots, as well as between the jumping and walking plots. And overall the plots are very distinct, making them easy to differentiate from each other.

In the 3D case, plots are again well distinct, and again they are really easy to differentiate from each other,they do not overlap that much. Just like in the 2D case, the jumping plot looks a bit different than walking and running plots; it looks like a spike, rather than an ellipse.

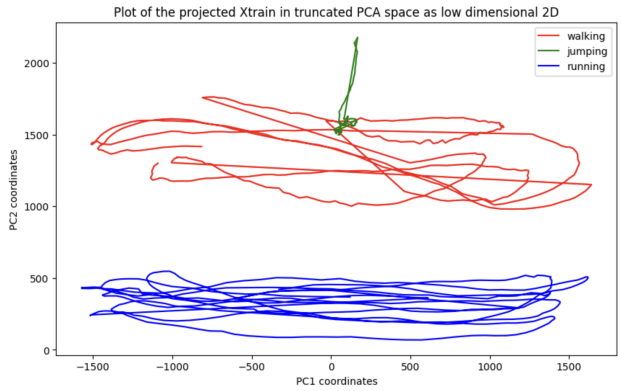

Figure 3: Projected Training Data in 2D Truncated Space

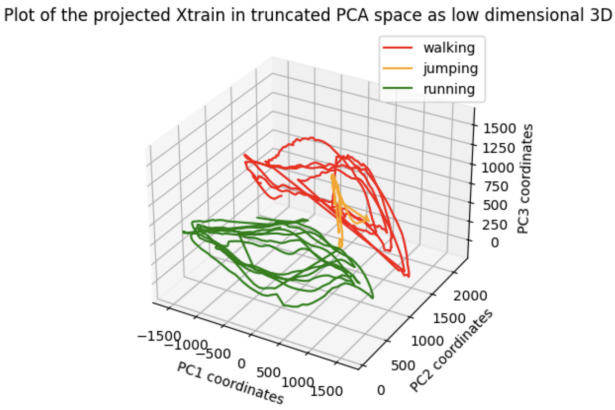

Figure 4:Projected Training Data in 3D Truncated Space

Employing the classification algorithm, each training dataset is categorized into either jumping, walking, or running for various k values in k-PCA truncation. The accuracy score functions as a metric to assess the precision of classification at each k value, representing the percentage of correct classifications made for the entire set of test or train data It is calculated based on the classifier's correct classification of different datasets, earning a positive point for each accurate classification. Figure 5 is the plot of Accuracy Score in percentages vs. first 20 k values in k-PCA Truncation created with the training data set, and Figure 6 is basically the same plot made with the Test Data Set.

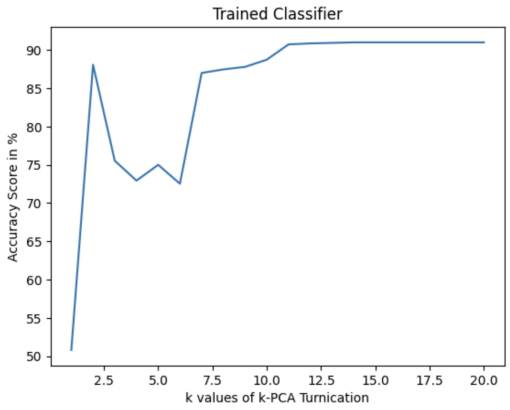

Figure 5:Projected Training Data in 3D Truncated Space

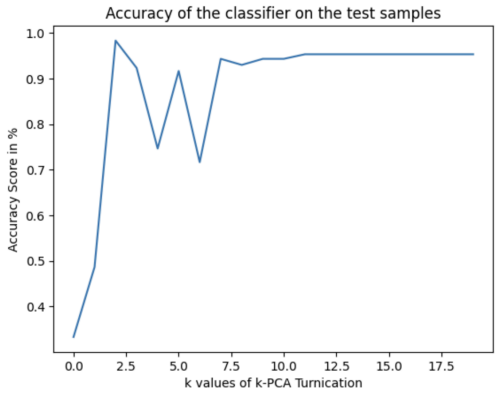

Figure 6:Projected Testing Data in 3D Truncated Space

Both Figure 5 and Figure 6 have similar shapes. As k values are increased the Accuracy percentage increases at a higher rate at small k values and the rate decreases as k value increases, reaching an asymptotic value at higher k values. This gives the impression that first k values have higher importance in the classification algorithm then the rest. In the Testing phase after 12th PCA mode, the Accuracy sits on an asymptote at %95 Accuracy rate and in the Training phase the Accuracy sits on an asymptote at %91 Accuracy after the 14th PCA mode, meaning we need at least 14 PCA modes to have a reliable Accuracy score and classification algorithm.

After 14 PCA modes, the accuracy of Training and Testing phases are similar and they are consistent. At lower PCA values they are not that much consistent in terms of Accuracy, they have some differences, some inconsistencies. This difference arises from training and or testing data being unable to capture every different scenario. This means either we need more training and test sets or better training sets that capture morte different scenarios. But overall the algorithm still works with really high accuracy percentages with enough PCA modes used. There is no overfitting.

Summary and Conclusion

I successfully built a projection of data sets to a lower dimension than the number of coordinates, visualized the movements and then based on it designed an algorithm that is able to recognize which one of three movements, jumping, running, walking, is humanoid robot doing in real time. I discussed the potential for further enhancement of the training dataset to capture a greater variety, thereby increasing the accuracy score of the trained classifier algorithm at testing, and decreasing the accuracy score differences between testing and training . But the overall accuracy scores were good enough and consistent, especially after 14 PCA modes. Thanks to PCA, we can construct a dependable classifier without the necessity to utilize every available mode, resulting in a more streamlined model for our algorithm.

Acknowledgements

I would like to thank my classmates Po, Shavey, Emoji for useful discussions and Professor Eli for his notes, lectures, Office Hours and code examples

References

[1] Eli Shlizerman (2024). 582 KutzBook. AMATH 582 Wi 24: Computational Methods For Data Analysis.