UNSUPERVISED LEARNING USING DIMENSIONAL REDUCTION IN MOVEMENT RECOGNITION

EDRIA LI

Applied Mathematics Department, University of Washington, Seattle, WA
yijueli@outlook.com

Abstract. We successfully projected the recordings of movements of humanoid robot OptimuS-VD joints to lower dimensions. We also visualized the movements and trained the data with supervised learning. Out test results showing a 95.33% accuracy in predicting the movements. We have implemented the training through `numpy` SVD and `sklearn`'s PCA package.

1. Introduction and Overview

We are given a humanoid robot OptimuS-VD which has built-in sensors that record the movements of 38 joints with rate of 60Hz. Each joints are represented with points in xyz coordinates. Five samples of each of the three movements (running, walking, jumping) are recorded for 1.4 seconds with 100 timesteps and saved as a matrix of 114 $\times$ 100. We are to build a projection of the recordings to a lower dimension and visualize the movements, and design an algorithm to recognize the movements. Test sets are provided.

The above-mentioned task can therefore be modeled into a dimension reduction problem and can be trained through supervised learning. We first bundled all the data in the 5 samples into a 114 $\times$ 1500 training matrix with each column represents one snapshot in time and each row represents the xyz coordinates.

We then apply the dimension reduction methods to lower the dimension of our training matrix into different numbers of PCA modes. We investigated the PCA modes using both SVD and `sklearn` PCA method. 2D and 3D data visualization were plotted and it can be clearly observed that different movements were clustered in different niches. Energy of different PCA modes in Frobenius norm was also investigated. We then established an algorithm using distance of joints to the centroid of movement to classify movements into 3 catagories (running, walking, jumping). We trained our labeled training set under supervised learning with an accuracy rate converged around 91.07% for PCA modes $k \geq 14$; and for provided data sets, we were able to recognize movements with an accuracy rate of 95.33% for PCA modes $k \geq 11$.

2. Theoretical Background

2.1. **Singular Value Decomposition.** The mathematical fundamentals regarding dimension reduction lies in Singular Value Decomposition (SVD). SVD is a factorization of a matrix into a number of constitutive components all of which have a specific meaning in applications [1]:

$$A_{m\times n} = U_{m\times m}\Sigma_{m\times n}V^{\top}_{n\times n} \tag{1}$$

where each column of U consists left singular vectors $\mathbf{u}_j$, columns of V consists right singular vectors $\mathbf{v}_j$, and the diagonal of Σ denotes positive singular values σ_i sequenced from upper left to lower right in order $\sigma_1 > \sigma_2 > \cdots > \sigma_r$ where $r = Rank(A)$.

Moreover, a reduced version of (1) called *reduced SVD* is prevalent in real world application as well:

$$A_{m\times n} = \widehat{U}_{m\times n}\widehat{\Sigma}_{n\times n}V^{\top}_{n\times n} \tag{2}$$

where dimension of left singular vector matrix $\widehat{U}$ and singular value matrix $\widehat{\Sigma}$ is now reduced while right singular vector matrix $V^{\top}$ stays the same. We will use this idea of reduced SVD equation (2) for our code implementation using SVD method.

2.2. **Principle Component Analysis.** Principle Component Analysis (PCA) is one alternative variant of the key applications of the SVD. The basic idea behind PCA is to dimensionally reduce a large data sets by projecting the large dataset into lower dimension along its principle axis through calculating covariance matrix under Gaussian normal distribution centered at zero $N(0, C_{\mathbf{x}})$:

$$C_{\mathbf{x}} = \mathrm{Cov}(\mathbf{X}) \approx \frac{1}{N-1}\mathbf{X}\cdot\mathbf{X}^{\top} \tag{3}$$

$$= \frac{1}{N-1}U\cdot\Sigma^2\cdot U^{\top} \tag{4}$$

where $C_{\mathbf{x}}$ denotes the covariance matrix to approximate data in $\mathbf{X}$ along principle axis. The eigenvalue decomposition shown above (3) has eigenvectors $\mathbf{u}_i$ same as left singular vectors of matrix $\mathbf{X}$. Additionally, eigenvalues λ_i of the covariance matrix is the square of singular value of $\mathbf{X}$: $\lambda_i = \sigma_i^2$. In furtherance of application, PCA projection can be very useful to represent data matrix $\mathbf{X}$ in a new basis of U (i.e. the left singular vectors as new basis) where the full projection is:

$$U^{\top}\cdot\mathbf{X} = \Sigma V^{\top}$$

the geometric meaning of the left side of the equation means the projection of original matrix into new PC space. And in order to lower the dimensions of $\mathbf{X}$, we can truncate the new basis by keeping the first k columns of U and setting all remaining columns to zero, i.e. U_k:

$$U_k^{\top}\cdot\mathbf{X} = \widetilde{\mathbf{X}} \tag{5}$$

k denotes the modes of PCA, where PC1 means to project data into 1 dimension along the most dominant variance direction. It is straightforward to apply $k = 2$ or $k = 3$ in data visualization where we project original data into 2 or 3 dimensions for better understanding the data structures.

2.3. **Energy and Frobenius Norms.** Given the properties of unitary matrix and SVD, we can easily derive Frobenius norm of a matrix through its singular values:

$$\|A\|_F = \|\Sigma\|_F$$

An important application of Frobenius is energy approximation: to see how much of the energy is being included in particular number of singular values where smaller singular values can be zeroized (or truncated). We will use the following equation for cumulative energy approximation to estimate how much energy do we still have in our representation of projected matrix:

$$\sum E_j = \sum_{k=1}^{j}\frac{\sigma_j^2}{E},\ E = \|A\|_F \tag{6}$$

3. Algorithm Implementation and Development

we used `Python` in which we import `numpy`, `PCA` from `sklearn.decomposition`, and `accuracy_score` from `sklearn.metrics` for computing; we also import `mplot3d` from `mpl_toolkits`, and `pyplot`, `animation` from `matplotlib` for plotting.

Here are the detailed algorithms for the six tasks completed as shown in Algorithm 1 to 6:

Algorithm 1 Task 1: Plot First 5 PCA Modes

```
1: Construct training set $X_{\text{train}}$ of size $114 \times 1500$.
2: Center the $X_{\text{train}}$ by
       $$\widehat{X_{\text{train}}} = X_{\text{train}} - \texttt{mean}(X_{\text{train}}).$$
3: Compute SVD on centered training set $\widehat{X_{\text{train}}}$
       $$\widehat{X_{\text{train}}} = U\Sigma V^\top.$$
4: Plot the first 5 columns of $U$.
```

Algorithm 2 Task 2: Cumulative Energy Ratio

```
1: for $j = 0, \ldots, 9$ do
2:     Truncate $\Sigma$ by setting $(j+1)$-th and on element to zero as $\widetilde{\Sigma}_j$
3:     Compute the power ratio by
           $$E_{\text{ratio}} = \frac{\|\widetilde{\Sigma}_j\|_F^2}{\|\Sigma\|_F^2}.$$
4:     Plot the ratio.
5: end for
```

Algorithm 3 Task 3: Trajectories of movements in 2/3D PCA space.

```
1: Compute PCA of $X_{\text{train}}^\top$ with 2/3 components as $X_{\text{proj}}^\top$.
2: Restore the result by transpose $X_{\text{proj}}$.
3: Plot trajectories of movements in 2/3D PCA space.
```

Algorithm 4 Task 4: Compute Centroids

```
1: Split the projected data $X_{\text{proj}}$ along the second axis into three pieces.
2: Compute the mean along the second axis to get centroids.
```

4. Computational Results

4.1. **Task 1. Plot First Five PCA modes.** We first load the training data files into one matrix X_{train} of dimension 114×1500 where each column correspons to each snapshot in time. After applying SVD to our orginal matrix, we plot the first 5 columns of U (i.e. $\mathbf{u_1}, \cdots, \mathbf{u_5}$) in Figure 1a, where we can see the magnitude of oscillation reflects value of elements in $\mathbf{u}_j$.

4.2. **Task 2. Cumulative Energy Ratio.** We tried out different k values of PCA modes and found out that the reduced-dimensional projected matrix $\widetilde{X}$ can preserve energy pretty well in a steep curve and flatten out after $k \geq 5$. We plot from $k = 1$ through $k = 10$ and printed cumulative energy ratio as compared to the energy of the original matrix. As shown in Figure 1b, we can find that $k = 2$ can approximate 72.66% X_{train} in the Frobenius norm; $k = 3$ can approximate 82.95% X_{train}; $k = 5$ can approximate 91.28% X_{train}; and $k = 7$ can approximate 95.40% X_{train}. We can infer from the data that the energy are mostly preserved even with very low dimensional representation. This indicates that in real world approximations, dimensional reduction has an overall benefit of lowering computing cost while at the same time preserve the most information needed for scientific analysis.

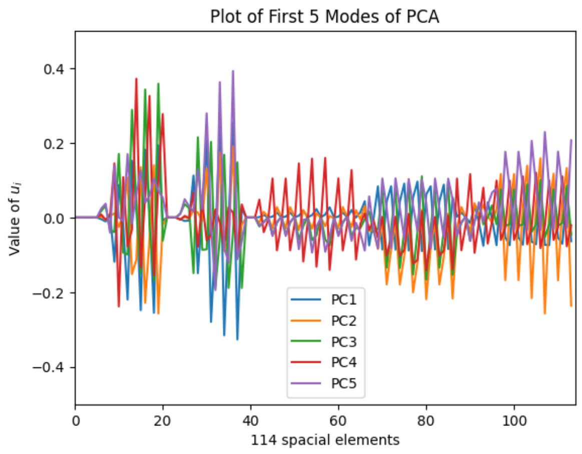
4.3. **Task 3. Plot Trajectories of Movements in 2D and 3D PCA Space.** We have implemented this task through the usage of `PCA` in package `sklearn` and get the following two-dimensional (where PCA modes $k = 2$) as shown in Figure 2a and three-dimensional ($k = 3$) graphs as shown

Algorithm 5 Task 5: Compute Trained Label and Accuracy

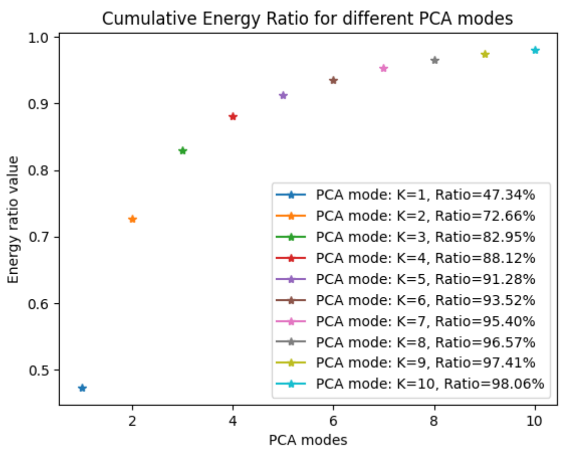
```
1: for $k = 0, \ldots, 19$ do
2:     Compute projected data $X^k_{\text{proj}}$ by PCA.
3:     Compute centroids by $X^k_{\text{proj}}$
4:     Compute distance from $X^k_{\text{proj}}$ to centroids.
5:     Compute the indices of minimal and assign indices to construst trained label.
6:     Compute accuracy again ground truth label.
7: end for
```

Algorithm 6 Task 6: Predict Test Labels

```
1: Construct test dataset.
2: Construct ground truth labels for test set.
3: for $k = 0, \ldots, 19$ do
4:     Compute projected data $X^k_{\text{proj}}$ by PCA on $X_{\text{train}}$.
5:     Compute centroids by $X^k_{\text{proj}}$
6:     Compute projected test data $X^{\text{test},k}_{\text{proj}}$ by PCA on $X_{\text{test}}$.
7:     Compute distance from $X^{\text{test},k}_{\text{proj}}$ to centroids.
8:     Compute the indices of minimal and assign indices to contrust trained label.
9:     Compute accuracy.
10: end for
```



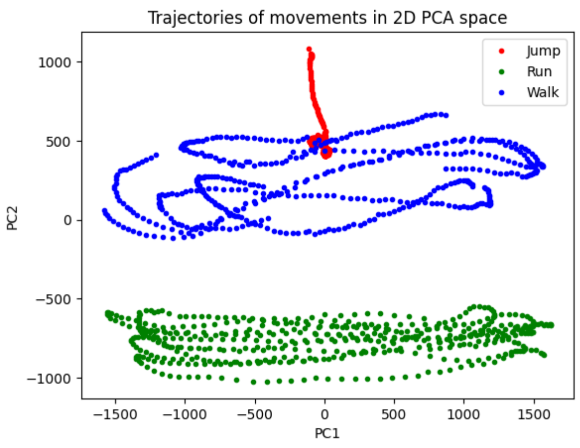
(A) First Five Modes of PCA.



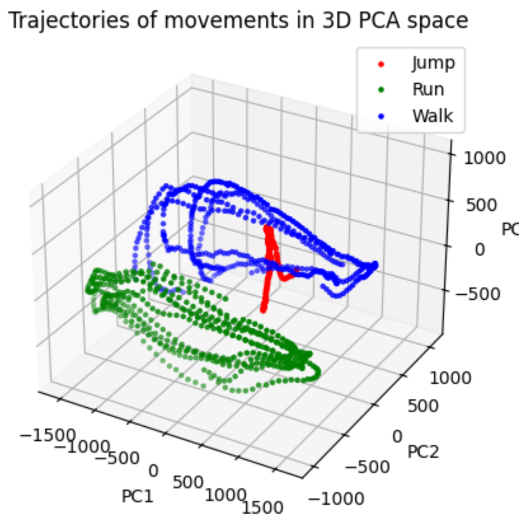
(B) Energy Ratio of 10 modes

in Figure 2b. As we can see, three different clusters of points are highlighted using different colors, where the red dots denotes jumping movements, blue dots denotes walking movements and green dots denotes running movements. The graphs meet our expectation that the spacial movement of jump are more likely to be limited in one coordinate; additionally, when compared the running and the walking graphs, we can tell that general cluster patterns are pretty similar albeit the clusters are located differently with running has a higher magnitude corresponding to faster speed (and "bigger" movement) of running than walking.

4.4. **Task 4. Compute Ground Truth and Centroids.** We generated the ground truth label in align with the sequence of training data that constructed matrix X_{train} as Y_{train}. We used discrete label 0 (jumping), 1 (running), and 2(walking) for classification purpose. We then computed the centroids of each movements. When $k = 2$ the centroids are: $(-23.89, 499.37)$



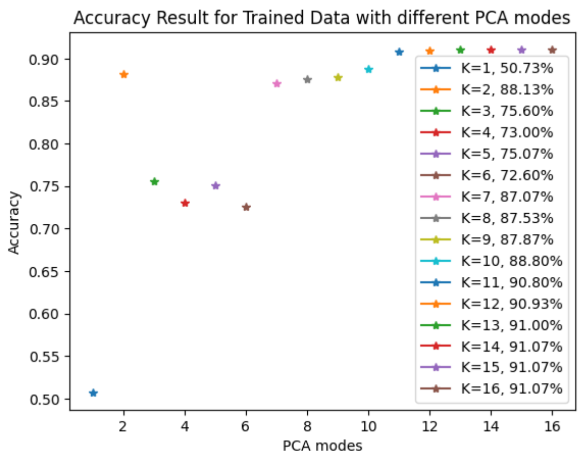
(A) Trajectories of movements in 2D PCA space.



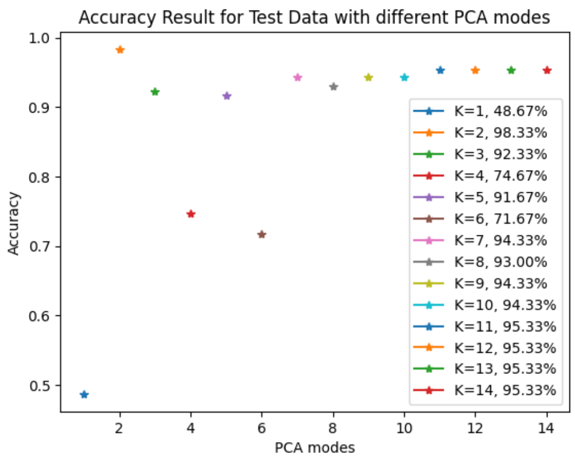
(B) Trajectories of movements in 3D PCA space.

for jumping, $(60.77, -752.72)$ for running, and $(-36.88, 253.35)$ for walking; when $k = 3$ the centroids are: $(-23.89, 499.37, -72.50)$ for jumping, $(60.77, -752.72, -103.41)$ for running, and $(-36.88, 253.35, 175.91)$ for walking, respectively.

4.5. **Task 5. Compute Trained Label and Calculate Training Accuracy.** We trained the training sets using different PCA modes and found out that after $k \geq 14$ the accuracy converge to 91.07% as shown in Figure 3a. We can observe that the accuracy rate does not increase monotonically, but have some curves for the initial values of k. This is normal and within expectation because fluctuations is pretty normal in low dimensions. And the accuracy rate cannot improve after certain PCA modes. One explanation maybe that there are certain overlaps of points from walking and jumping that cannot be further distinguished through mere centroid method.



(A) Accuracy Result for Training Set



(B) Accuracy Result for Test Set

4.6. **Task 6. Predict Test Labels and Calculate Test Accuracy.** After training, we then compared our test sets data with the training centroids to predict the test labels. We label the test sets using different PCA modes and found out that after $k \geq 11$ the accuracy converge to 95.33% as shown in Figure 3b. Similar trends of data (non-monotonic and could not reach 100% accuracy) can still be observed.

5. Summary and Conclusions

To predict the movement classification of the humanoid robot, we used SVD and truncated PCA projection to reduce the dimensions of our recording data. We completed the following steps: (1) established the right dimensions of our training data; (2) visualized the first five PCA modes for better understanding of different modes; (3) investigated cumulative energy ratio to see how much energy (Frobenius norm) are still preserved by dimensional reduction; (4) visualization of 2D and 3D trajectories of movements in PCA space; (5) computed ground truth table used for supervised learning; (6) trained our data using centroids computed for each movement; (7) calculated the accuracy for both training set and the test set. We discussed our findings for preserved energy and accuracy result for training set. We observed similar accuracy trends for our test set. In order to increase our overall accuracy rate for classification, alternative classifiers maybe needed.

Acknowledgements

The author is thankful to Prof. Eli for useful discussions about SVD, PCA projections, data structures, and the perspective to perceive norm as energy.

References

[1] J. Kutz. *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data.* Data-driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data. OUP Oxford, 2013.