COGS 118B Final Project

Jialu Sui jisui@ucsd.edu
Qixuan Ma q5ma@ucsd.edu
Yiyang Cao yicao@ucsd.edu
Lehan Li l8li@ucsd.edu

Abstract

As wearable devices together with smartphones became increasingly involved in human's life over the past few decades, an enormous amount of data has been generated through daily engagement with such devices. This project seeks to classify human daily behavior through data collected in waist-mounted smartphones with embedded inertial sensors. Unsupervised learning algorithms are used including PCA and K Means Clustering to conduct clustering analysis. Additional supervised learning (Logistic Regression and Random Forest Classifier) are conducted and are used as extra parallel comparison analysis.

1 Introduction

Usage of portal electronic devices constitutes a large proportion of daily activity. Millions of data is generated through human engagement with smartphones, patterns of human activities are embedded inside the seemingly messy datasets. This project seeks to extract the useful features from the enormous dataset, and accomplish human activity recognition based on data collected by waist-attached smartphones. Total of 30 volunteers within an age bracket of 19-48 years are participants in this research. Sensor signals captured by the smartphones are preprocessed by applying noise filters and sampled in a fixed-width sliding window. Total of 6 activities are included in the dataset: walking, walking upstairs, walking downstairs, sitting, standing and laying. Human activity recognition is significant since it could be deployed to measure the exercise of the users, and to create formatted exercise reports. It provides a convenient way to keep track and organize daily activities, and could be used as references to create personalized health tips.
Unsupervised learning algorithms are used to conduct data analysis, including PCA and Kmeans (both implemented from scratch and implemented using sklearns). Supervised learning models such as Logistic Regression and Random Forest Classifier are also implemented to conduct parallel comparison analysis. Various metrics (AMI Scores, Homogeneity Scores, Completeness Scores, V Measure Scores, FM Scores, Silhouette Scores and Adjusted Accuracy) are used to measure the performance of models to analyze the effectiveness of PCA.

2 Related Work

Previous research is conducted regarding human activity recognition in Machine Learning. Paola Ariza Colpas [1] accomplished ADL recognition using multiple approaches with different dataset, including Partitional Method (K means), Kernel-Based methods (Kernel K means), and Spectral Methods (Spectral Clustering). Wider range of activities and multiple dataset are used in his research, and parallel comparison is analyzed. Omran and his colleagues were devoted to conduct analysis on different representation clustering techniques, including the heuristic approaches to the clustering problem. [2] Besides, evolutionary models are used for Human Activity Recognition. Holland contributed to the development of computer algorithms that seek to mimic genetic evolution and examine the interactions of small entities that lead to complex scenarios on a larger scale. [3]

3 Method

Two types of algorithms are used: unsupervised learning and supervised learning. Within unsupervised learning, two different types of implementations are used: implemented from scratch, implementation based on existing libraries (scikit-learn).

3.1 Unsupervised Learning

3.1.1 Custom Implementation of PCA

In order to reduce dimensionality of our dataset and therefore visualize the clusters, we use PCA to reduce dimensionality.For implementing PCA, we adopted the approach introduced in lecture. Since our dataset has significantly more entries than attributes, we did not use the transpose trick introduced in lecture to decrease computational complexity and calculating covariance matrix directly from the mean-zeroed data gives promising computation complexity.
Since our dataset is currently of shape (10299 * 561), where rows indicate data entries and columns indicate attributes, we need to first transpose this dataset to make each column a data entries and each row a unique attribute in order to calculate the covariance matrix in an efficient way. In order to apply PCA on our dataset to reduce dimensionality, we first compute the mean and subtract from all the data, and then calculate the covariance matrix of the mean-zeroed data and find its eigenvectors and eigenvalues. And then we normalize the eigenvectors to unit vectors. Furthermore, we sorted the eigenvectors in order of decreasing eigenvalues and put the sorted eigenvectors as columns in matrix V. Then, we calculate the first n principal components by dot-producing the first n sorted eigenvectors with the mean-zeroed dataset.

3.1.2 Custom implementation of K-Means

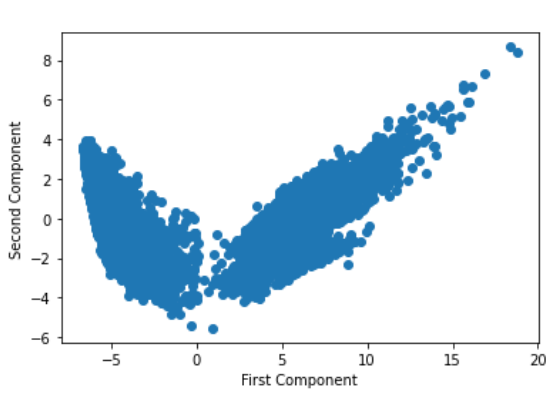
Besides implementing PCA, we also self-implemented K-means to cluster the dataset. We adopted a classical implementation of K-means introduced in lecture as well as in Dr. Bishop's book. We start with some initial, random guess of the cluster centroids. Then, we keep running the algorithm for iterations to ensure convergence of the clustering. In each iteration, we first calculate the squared distance of each dataset to each cluster centroid, and then use this information to determine the "responsibility" of each cluster centroid, where a data point x is assigned to centroid k if and only if it is closest to cluster centroid k than others in squared distance. After assigning responsibilities, we then update the cluster centroids using the new cluster assignments where each cluster centroid is updated to be the mean of all data points that are closest to it. We would keep iterating until the changes in cluster centroids are minuscule. At the very end of this algorithm, we could cluster the entire dataset into k distinct clusters. For the purpose of visualization, we first apply PCA on the original dataset, and then do the clustering based on the first 2 components of the PCA.

3.1.3 Library Methods

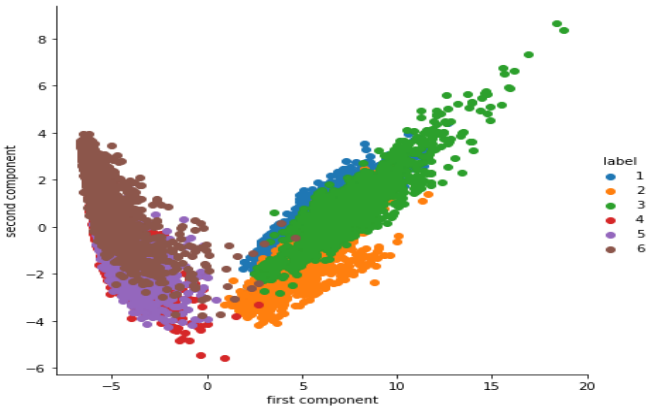
We used the PCA() function that comes with the scikit-learn package. We then called the fit_transform method to transform our predictors, shaped 10299 by 561 into a 10299 by 2 dataset where the columns are the two principal components of the original dataset. For Kmeans, we then fitted a KMeans clustering algorithm with k = 6 on the 2 principal components extracted from the original dataset as described in the last section.

3.2 Supervised Learning

We decided to use the three most common sklearn supervised learning algorithms to compare the results with both sklearn Kmeans and self-implemented Kmeans. The three classification algorithms we chose are Random Forest, Logistic Regression, and K-Nearest Neighbor.
Our dataset has already been split into training data x_train (7352 * 561) and y_train, and testing data x_test (2947 * 561) and y_test. Take logistic regression as an example, we create a classifier using LogisticRegression(), fit model on x_train and y_train data using fit() method, and predict the labels for x_test data using predict() method. To evaluate the model, we used the accuracy_score method to get the accuracy of each model, and printed the overall report by using the classification_report method which included the metrics like f1-score, accuracy, and precision. For the purpose of comparison, we use these statistics to compare the results of the classification models and K-Means.



(a) scikit-learn K-Means clustering on PCA result



(b) scikit-learn K-Means clustering on PCA result

Figure 1: A figure with two subfigures

4 Results

4.1 Unsupervised Learning

4.1.1 PCA using custom implementation

Our custom PCA gives promising results. Using PCA coded by ourselves, we reduced the dimensionality of the dataset from 561 to 2. With proper visualization which is shown below, we could see that the PCA result preserved the 6 clusters pretty well. However, since our original dimensionality is too large, the first 2 components of PCA could only explain 67 percent of the variance in the original dataset. If we want to preserve at least 85 percent of the variance, we can only reduce dimensionality from 561 to 20, which is obviously not ideal for the purpose of visualization. Therefore, in order to visualize the clusters, we decided to trade variance for more dimensionality reduction. [Figure 1.2]

4.1.2 K-Means clustering using custom implementation

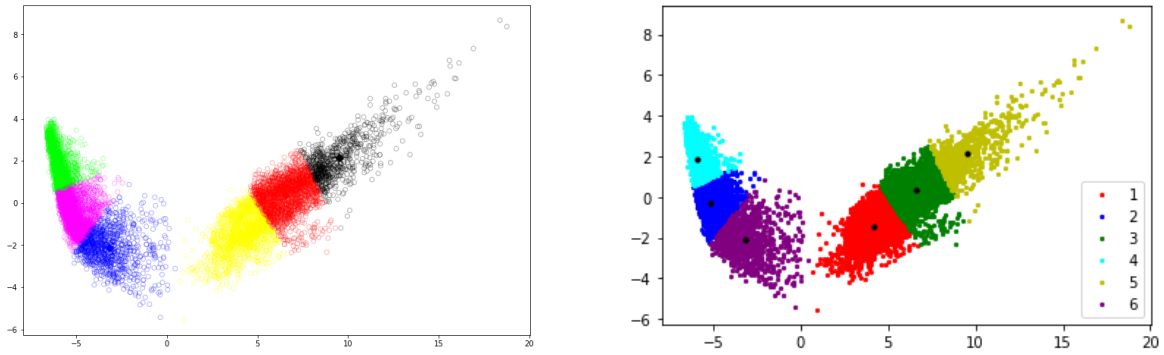
Our custom K-means also clustered the data into 6 clusterings. However, the clusterings detected by our K-means do not seem to reflect the true clusterings in the original dataset. We clustered based on the first 2 components of PCA. Since K-means clusters data points based on distances, it could no longer recognize the correct clusterings after we reduce the dimensionality from 561 to 2. Compared to the visualization of true clusterings after PCA, the result of the K-means simply reflects some blocks in the scattering, which does not imply the true clusterings in the dataset at all. [Figure 2]

4.1.3 PCA and K-Means using scikit-learn

First, we performed a PCA on the original dataset and ended up with a 10299 by 2 dataframe. We plotted the value of the first and the second principal components for each of the data points as x and y values in a scatter plot [Figure 3]. Following the step above, we fitted a K-Means clustering on our dimensionality reduced dataset and assigned a color to each of the predicted labels. [Figure 4]

4.2 Comparison between scikit-learn and custom implementation

Firstly in terms of PCA. We can see that this overlapped perfectly with Fig. 1, which assures that our custom implementation of PCA is correct, and will provide a baseline with which we can fit a K-Means Clustering method later on.
Then we can compare the results of the scikit-learn K-Means clustering against our custom implementation. Just as the graphs for PCA, the two clustering result plots look identical besides the difference in colors. With this result, we will assume that the performance (in terms of clustering result and not computational time) of our custom implementation is equal to the implementation provided by the scikit-learn library. The performance metrics discussed in the following sections will be based on the clustering results of the scikit-learn library implementation.



(a) custom K-Means clustering on PCA result

(b) scikit-learn K-Means clustering on PCA result

Figure 2: A figure with two subfigures

4.3 Comparison between model performance before and after PCA

Table 1: K-Means model performance before and after PCA

Score for each category by metric		
Metric	Before PCA	After PCA
Adjusted Mutual Information Score	**0.588**	0.458
Homogeneity Score	**0.579**	0.451
Completeness Score	**0.598**	0.466
V Measure Score	**0.589**	0.458
Fowlkes Mallows Score	**0.557**	0.422
Silhouette Score	0.132	**0.443**
Adjusted Accuracy	**0.602**	0.502

We can see that the performance for K-Means clustering is higher across the board when we fit the K-Means on the original dataset rather than the dimensionality reduced dataset after PCA. The only exception is for the Silhouette Score, which measures the separation between clusters, which does not have significant implications in terms of performance since the 2 K Means clustering was performed in 2 different dimensions.

4.4 Comparison between K-Means and supervised learning algorithms

Table 2: K-Means and LogReg model performance

Score for each model by metric		
Metric	K-Means	Logistic Regression
Accuracy	0.19	**0.96**
F1-Score	0.16	**0.96**
Precision	0.14	**0.96**
Recall	0.19	**0.96**

Technically, it is unreasonable to compare supervised learning algorithms with unsupervised learning algorithms, but here we try to explore the difference on the performance of two types of models on the same dataset (we do not use y on PCA and K-means model, making it latent).
It is hard to compare the classification model with custom implemented Kmeans because it is hard to evaluate our algorithm using statistics such as accuracy, f1-score, and precision. However, from the visualization of K-Means clustering, we see that it does not imply a true clustering. Therefore, if we only compare the clustering performance for these two types of model, the classification model performs much better than Kmeans in this case.
Comparing the classification model with scikit-learn Kmeans is more straightforward. From the table above, we could see that all metrics including accuracy, f1-score, precision and recall show much higher performance on classification models than on K-means clustering. This also shows how important the latent variable which

is the labels here could be on the prediction of the dataset. If we know what data belongs to what group, we could use supervised learning to welly predict the other data which does not have the labels but might come from the same population. However, we cannot conclude that classification algorithms are much better than K-means model, since our original dataset has clustering labels while in the real circumstances, dataset usually does not have these labels, in this case, Kmeans and other clustering algorithms will shine.

5 Discussion

In this project, we see that the K-means model doesn't perform very well on assigning clusters after PCA. In our project, for better visualization of the clusterings, we first perform PCA to reduce dimensionality, and then perform K-means on the first 2 PCA components. However, since K-means cluster data based on distances, when data points are compressed from high dimensions to 2 dimensions, different clusters highly overlap with each other in the 2-d world. Thus, K-means could not correctly identify the clusters in the original dataset since distances in this 2 dimension does not reflect true distances in the original dataset. If we had more time, we could try other clustering methods such as the Mixture of Gaussians to see whether it would end up with better clusterings. The reason we think MOG might would work better because from class we know it is suitable for the cluster which is elongated, and here, from the plot of PCA, our clusters are somehow elongated, so MOG might perform better.
Moreover, we lost almost 30 percent variance after reducing dimensionality from 561 to 2 for visualizing the clusters in 2 dimensions. PCA could be dangerous if we lost too much variance in the original data. Thus, in order to preserve better accuracy and variance, we have to avoid excessive information loss due to dimensionality reduction. Luckily, our dataset does not lose too much information after PCA, but we do think it's necessary to preserve at least 85 percent of the variance. Thus, if we have more time, we would not trade variance for better visualization. Instead, we would keep the first 20 PCA component (which could preserve 85 percent of the overall variance), and then try to perform some clustering algorithm on the dimension-reduced data to identify different clusters. This should give us better metrics results (e.g accuracy score) since we are now preserving more variance in the dataset. If we really want to visualize the result of the clustering, we could instead visualize in 3-dimensional space using the first 3 PCA components. This could help us better separating the data clusters and potentially gives more promising results.

6 Author Contributions

1. Jialu Sui: Custom Implementation of PCA and K-means, Discussion section
2. Yiyang Cao: Supervised Learning Method, Comparison between K-means and Supervised learning, Discussion
3. Qixuan "Harrison" Ma: scikit-learn implementation of PCA and K-Means, comparison between scikit-learn and custom implementation, comparison between pre-PCA and post-PCA performance.
4. Lehan Li: Introduction and Literature Review

7 Appendix

All code and datasets related to this project can be found in this Github repository: https://github.com/Harrison-Q-Ma/COGS118B_Final_Project

8 Reference

1. Ariza Colpas P., Vicario E., De-La-Hoz-Franco E., Pineres-Melo M., Oviedo-Carrascal A., Patara F. Unsupervised human activity recognition using the clustering approach: A review. Sensors. 2020;20:2702. doi: 10.3390/s20092702.
2. Omran M.G., Engelbrecht A.P., Salman A. An overview of clustering methods. Intell. Data Anal. 2007;11:583–605. doi: 10.3233/IDA-2007-11602.
3. Holland J.H. Hidden Orderhow Adaptation Builds Complexity. Helix Books; Totowa, NJ, USA: 1995.

9 Acknowledgements

1. In implementing custom PCA, the eigsort and normc function is from hw4 starter code.

2. Code for calculating explained_variance in custom PCA is from https://towardsdatascience.com/principal-component-analysis-pca-from-scratch-in-python-7f3e2a540c51
3. Codes for implementing K-means are from Jialu's homework 2 answers with revision.