Multi-modal Unsupervised Classification on Activities of Daily Living (ADL) Dataset

Justin Dannemiller
Advanced Telerobotics Research Lab
Kent State University
Kent, Ohio, United States
jdannem6@kent.edu

Keith Martin
Advanced Telerobotics Research Lab
Kent State University
Kent, Ohio, United States
kmachina@kent.edu

Bailey Wimer
Advanced Telerobotics Research Lab
Kent State University
Kent, Ohio, United States
bwimer3@kent.edu

***Abstract*—Supervised learning is a widely popular machine learning technique that has achieved remarkable success in wide-ranging applications, such as cancer cell identification and spam detection. Despite it's success, supervised learning imparts a strong requirement that the underlying training data be well structured and labeled. In many domains, however, this label distribution is not available or is otherwise very challenging to achieve. In such cases, unsupervised offers a powerful alternative. In this paper, we seek to prove the effectiveness of unsupervised learning by applying it to a data-set which tracks the daily activities of people around their houses. To do so, we have implemented and tested three different unsupervised clustering models: K-Means, Gaussian Mixture Model, and Mean shift. By examining the results of the data, we are able to conclude the effectiveness of each of the models by measure of both accuracy compared to the labeled data as well as visualizations of clusters.**

***Index Terms*—Machine Learning, Unsupervised Learning, K-Means, Gaussian Mixture Model, Activity Model**

I. Introduction

Unsupervised learning, which is defined as learning without an examples or labels, has continued to become more popular in the recent past for a couple of reasons. First, due to the enormous amount of data generated in a single day, it is becoming more and more difficult to label data. Recent statistics depict that 463 exabytes of data are produced everyday; this amount is simply beyond imagination. Data labelling tasks requires human intervention to ensure top notch accuracy, which is an expensive task. Second, there's always an element of error when there is human intervention meaning that the results of the labelling task might not be fully accurate. Also, data labelling is extremely time consuming. For these reasons, unsupervised learning techniques have grown in popularity because it is possible to gather predictions and insights even with this kind of data. Some of the popular use cases of unsupervised learning include the following: recommendation engines: on an e-commerce platforms, recommendation engines are used to give suggestion to customers on what to buy based previous data [1].

Another use case is customer segmentation. In this approach, customers are grouped based on their behaviour such as buying trends [2], [3]. The main aim of doing this is to study each customer group individually and understand how to target each one of them with the intention of increasing revenue. Unsupervised learning is also applied in data mining to uncover patterns and trends in data [4], [5]. This particular uses case of unsupervised learning has also grown in popularity a great deal in the recent past. This paper will demonstrate various unsupervised methods and discuss how disparate unsupervised learning algorithms were harnessed to gain insights from data based on activities of daily living. This report will also convey the results of the experiments performed during this study.

II. Methodology

A. Data Preprocessing

1) Data Handling: In order to properly modify and utilize the data from the Ordonez dataset, it must first be converted into a format parsable by Python. The chosen approach to handle this data was the Pandas library, using their data frame feature. First, the column names are specified before the files are read due to inconsistent column labeling across the ".txt" files of the dataset. Then, the files containing each person's ADL and Sensor data is separately imported into its own data-frame using the corresponding columns. After the data has been initially imported, it is then handled through data frames for the duration of its existence within the program. At any point in the program where the form of the data is significantly modified, it is then saved into a corresponding ".csv" file of a more standardized format.

2) Feature Extraction: When attempting to cluster data, one important step is manual feature extraction. By manually extracting features, the model can be provided with important context to more accurately cluster the data. In the system, the feature extraction comes in the form of the time-based attributes. Specifically, the "Start Day," "Start Time," "End Day," and "End Time" attributes are converted into attributes containing the hour of the day, day of the week, and duration of the activity. Table I shows the structure of the data before feature extraction, and Table II shows the structure after its applied.

3) Labelling Data: The data from the ADL dataset includes 10 different activities of daily life. The sensor data is combined with the ADL data by comparing the time at which sensors were activated with the time the subject performed actions. By doing so, each sensor activation is labeled with the name of

TABLE I: Data Before Feature Extraction

Start Date	Start Time	End Date	End Time	Location	Type	Place	Person
2011-11-28	02:27:59	2011-11-28	10:18:11	Bed	Pressure	Bedroom	Person A
2011-11-28	10:21:24	2011-11-28	10:21:31	Cabinet	Magnetic	Bathroom	Person A
2011-11-28	10:21:44	2011-11-28	10:23:31	Basin	PIR	Bathroom	Person A

TABLE II: Data After Feature Extraction

Duration	Time of Day	Day of Week	Location	Type	Place	Person
7:50:12	2	Monday	Bed	Pressure	Bedroom	Person A
0:00:07	10	Monday	Cabinet	Magnetic	Bathroom	Person A
0:01:47	10	Monday	Basin	PIR	Bathroom	Person A

the corresponding activity (such as “Toileting” or “Breakfast”). In any case where the time of a sensor activation did not correspond to an activity occurring, the data was instead labeled as “No Activity”. This results in a final total of 11 classes. The merging of the data sets was important since it aided in interpretation of model results. The predicted class label and the actual class label were juxtaposed in order to efficiently compare.

B. Encoding

After the manual pre-processing was finished, the next step was to reshape the data is encoding. In order to make the data processable by the system, it is important to convert all strings values to numbers. To do this we use a method known as ”One Hot Encoding” in which each unique option for an attribute is turned into an attribute of its own. The attribute that applies to each entry is given a value of 1 while all the other attributes in that category are given values of 0. For example, the ”Day of Week” attribute would be converted into seven different attributes (Monday through Sunday). All events that occurred on a Monday would have a value of 1 for Monday and 0 for all other days. This process is also applied to: ”Location,” ”Type,” ”Place,” and “Person.”

One attribute that is unique in this dataset is the ”Hour of Day” attribute. This attribute is interesting because hour 24 is closer to hour 1 than it is to hour 22, which, while intuitive to a human, is not intuitive to a computer. In order to resolve this issue, cyclic feature encoding was performed. We encoded ”Hour of Day” into ”Cosine Hour” and ”Sine Hour.” These attributes apply the periodic cosine and sine functions to improve the systems understanding of the relation of cyclical nature of hours in a day.

After applying ”One Hot Encoding” and normalizing, the dataset had a total of 35 attributes. In order to reduce this number, we next applied an algorithm known as Principal Component Analysis (PCA). PCA analyzes the covariance of the attributes to determine correlations in the data. From there, PCA is able to extract a smaller number of new features that create a holistic view of the dataset. In our case, PCA reduced the number of features from 35 to 18.

C. Clustering

The pre-processed and encoded data is grouped into 11 clusters, aligning with the 11 aforementioned classes. In order to do so, we implemented three clustering algorithms. The first algorithm, K-Means, which begins by randomly identifying seed clusters, examines the distances between points using euclidean distance metric to identify clusters for each point and iterative recalculates the centroid points till all points are fully clustered. The next method which we tested was the mean shift algorithm. The meanshift algorithm is a hill climbing like clustering algorithm, it begins by randomly identifying a centroid and gradually shifts the centroid, along with the specified radius of this centroid (sliding window), in the direction of the denser regions. The algorithm being a center point based clustering algorithm purposes to find an ideal center point that can be used to cluster all the other points. The sliding window continuously shifts till it cannot shift anymore in the direction of a denser region.

The third algorithm implemented uses Gaussian Mixture Models. This method randomly initializes a Gaussian distribution for each cluster and tunes them to generate the probability that a certain data point belongs in a cluster.

However, one issue persistent across all unsupervised learning is identifying which cluster relates to each of the true classes. To remedy this problem, we can compare the true classifications of the data in each cluster to determine the class that each cluster likely represents. Fig. 1 shows an example scenario involving two classes and two corresponding clusters. The figure also shows 10 points of data from each of the classes. By examining the amount of known data from each class in each cluster, we can see that the top (orange) cluster likely corresponds to Class B and the bottom (green) cluster represents Class A.

D. Hyper parameter tuning

This section will go over the hyper parameters that were used to tune each model to get optimal results.

1) KMeans Clustering: The following is a list of hyper parameters that were implemented: n_init, init, random state, max_iter, and algorithm. n_init was set to auto to automatically determine the number of times the algorithm will run on each cluster centroid initiated. random_state was set to 1234 enable reproduction of the algorithm results when running the algorithm in different during different instances. max_iter was set to 1000 to enable the model to run 1000 times in every single run. The default value is 300. 1000 was high to enable the model to run for longer. The algorithm parameter was

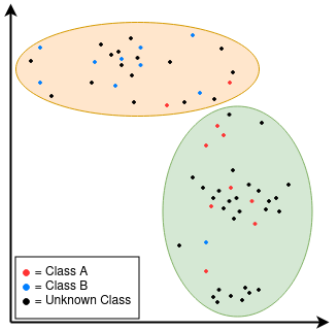

Fig. 1: Two unknown clusters containing many data points with unknown labels. 20 data points with known classes are shown, 10 from each class. These points allow us to identify the clusters.

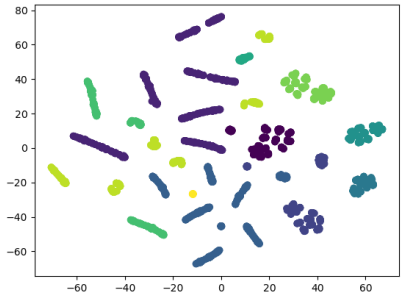

Fig. 2: A 2D representation of the clustering accomplished by the K-Means algorithm.

set to elkan which is the best model due to its fast speed of convergence.

2) MeanShift Clustering: The following is a list of hyper parameter that were implemented: bandwidth, cluster_all, bin_seeding, max_iter. Bandwidth was set to 6. The bandwidth value indirectly determines the number of clusters that will identified. Its is not direct like in KMeans, in mean shift algorithm, a smaller bandwidth value results in a larger number of clusters identified while a larger bandwidth value results in a smaller number of clusters identified. Unlike in KMeans, the number of clusters to be identified by mean shift algorithm is purely determined by the model but guided by parameters such as bandwidth. Cluster_all parameter was set to true to enable the model to cluster every single point plotted. Points that fall a bit far off still get assigned the closest cluster. bin_seeding parameter was set to True to speed up convergence because when the bin_seeding is True, the model doesn't try to initialize too many seed points which is time consuming. max_iter was set to 1000 to enable the model train for a bit longer with the intention of improving results.

3) Gaussian Mixture Model Clustering: The following is a list of hyper parameter that were implemented: n_components, random_state, warm_start. n_components was set to 11. Just like in mean shift clustering, a gaussian mixture model determines the number of clusters that are there, it doesn't not take this information from the program, like in KMeans. However, the parameter named n_components has a direct influence in determining the clusters because it represents the possible number of distributions that exist under the one big probability distribution of the large dataset. This could be indirectly translated as the number of possible clusters but not in all cases. random_state was set to 42. The number 42 enables reproducability of results when running the model during different instances. warm_start was set to True enables reuse of the model from previous training instances.

III. Results

For the purpose of these tests, all three models have been implemented in Python using the "scikit-learn" package. We also utilized the "pandas" package for data manipulation and "matplotlib" and "plotly" for data visualization. The following section will explore the results of the three models including the accuracy when compared to the labeled data as well as the visualized clusters.

In order to properly visualize the clusters, the pre-processed and encoded data was passed through a t-Distributed Stochastic Neighbor Embedding (t-SNE). This algorithm is used to drastically decrease the dimensionality of data to allow for improved data visualization and exploration. For our purposes, we utilized t-SNE to create both a 2D and 3D projection of our clustered data, allowing for the creation of visualizations like Fig. 2 and Fig. 3 respectively.

A. K-Means

The K-Means model was implemented using the hyper parameters mentioned in section II-D1. After training, the clusters produced can be seen in a 2D visualization in Fig. 2, which was produced using the aforementioned t-SNE model. The 2D clusters look reasonable, however, when the visualization is expanded to 3D (as shown in Fig. 3), the clustering becomes more clear. The confusion matrix in Table III shows the performance of the K-Means model compared to the original activity labels of the data. By examining the diagonal of the confusion matrix, we can calculate that K-Means resulted in an accuracy of 36.89%.

B. Mean Shift

When calculating the accuracy of the mean shift model, it results in an accuracy of 49.84%. This is the highest accuracy of the three models, however the number is very deceiving. Upon further examination, it becomes clear that the clustering was very inaccurate, which can be seen in both Fig. 4 and Fig. 5. The higher accuracy comes from the clustering pattern of this model, which resulted in only seven clusters, with one cluster containing over 78% of the data. This cluster was most closely associated with the "No Activity" class that much of the data-set is classified as. This created an artificially high accuracy for the mean shift model.

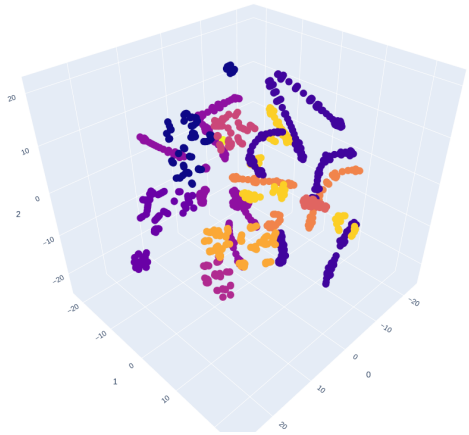

Fig. 3: A 3D visualization of the clustering results from the K-Means model.

TABLE III: K-Means Confusion Matrix

149	26	13	5	0	2	0	4	0	22	0
27	328	14	209	17	64	5	146	19	208	0
0	22	74	78	0	0	0	7	0	5	4
0	22	53	83	1	0	0	15	0	3	0
0	10	0	4	87	0	0	3	0	2	0
19	31	1	5	0	72	0	5	0	17	0
0	2	0	0	2	0	40	9	0	38	0
0	38	61	52	0	0	0	22	0	0	9
2	207	2	59	2	0	0	38	175	18	0
0	12	15	21	0	0	0	11	0	3	0
24	0	0	0	0	0	0	0	0	0	0

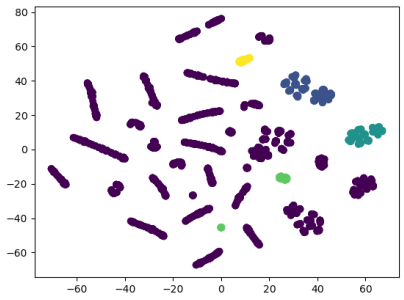

Fig. 4: A 2D representation of the clustering accomplished by the Mean Shift algorithm.

C. GMM

The final model, Gaussian Mixture Model, had the overall best performance. The clusters shown in Fig. 6 are noticeably more homogeneous than those in Fig. 2 or Fig. 4. The clusters

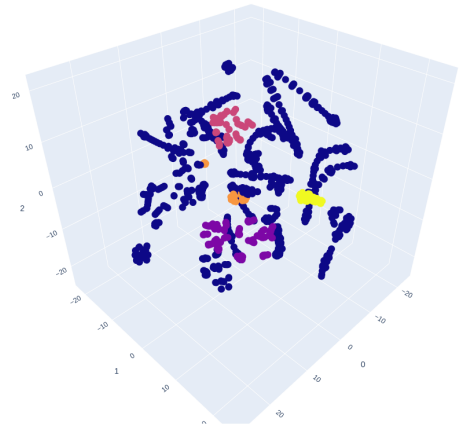

Fig. 5: A 3D visualization of the clustering results from the Mean Shift model.

TABLE IV: Mean Shift Confusion Matrix

926	19	64	17	2	5	4	0	0	0	0
326	175	0	2	0	0	0	0	0	0	0
78	0	72	0	0	0	0	0	0	0	0
19	0	0	87	0	0	0	0	0	0	0
144	0	0	0	32	0	6	0	0	0	0
49	0	0	2	0	40	0	0	0	0	0
137	0	0	0	18	0	35	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0
53	0	0	0	4	0	5	0	0	0	0
219	0	2	0	0	0	0	0	0	0	0
165	0	0	1	5	0	6	0	0	0	0

only look better in three dimensions, as shown in Fig. 7. The clustering of the model resulted in a 40.14% accuracy, which, while lower than mean shift's accuracy, appears to be a better overall estimator of the dataset. This can especially be seen when comparing GMM's confusion matrix (Fig. V) to mean shift's (Fig. IV).

IV. Discussion

When comparing the three models, it is clear which model had the worst performance on the data-set. That model is mean shift. Although it had the highest accuracy, when taking a closer look at the structure of the clusters, it can be seen that the accuracy number is inflated significantly. Not only did the model result in only seven cluster when there should have been eleven, it also had one cluster that contained over 78% of the data from the data-set. However, with the other two models, the performance was much more similar.

The clusters of both K-Means and GMM were visibly more well-defined and logical compared to the data-set. This is

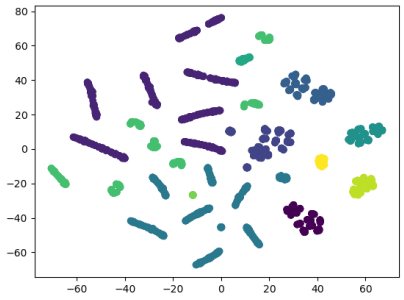

Fig. 6: A 2D representation of the clustering accomplished by the GMM algorithm.

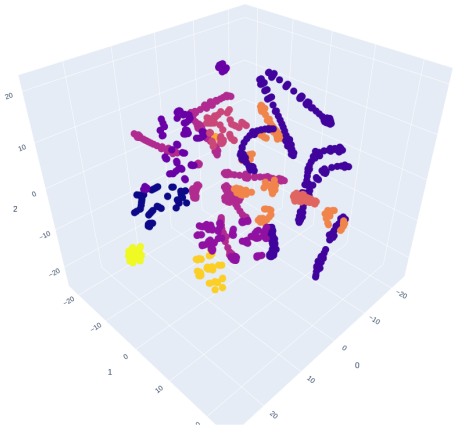

Fig. 7: A 3D visualization of the clustering results from the GMM model.

likely due to the nature of the problem at hand. This data-set is a labeled data-set, meaning there is a known number of expected clusters before training begins. As such, this problem benefited from a defined number of clusters, as in K-Means and GMM. Unfortunately, both models resulted in a lower-than-expected accuracy.

In order to attempt to resolve the accuracy issues, we took several steps, some of which were effective and others were not. As mentioned in Section II-D, we put a lot of effort into improving the performance of the models through hyper-parameter tuning. This did increase the performance of the models substantially compared to our first attempts. Another variable which we attempted to leverage to improve the models' performance was the number of features extracted by PCA. We initially had the feature count set to eight, but found that this resulted in a cumulative explained variance of only around 40%, meaning it was not generating a good representation of the data. By adjusting this number, we ended up settling on 18 features, resulting in a 100% explained variance.

One part of this problem which was not clearly defined was the way in which to handle sensor activations that are not related to any specific activity. In order to test the effects

TABLE V: GMM Confusion Matrix

56	22	0	0	84	0	0	6	4	0	18
10	386	29	19	263	64	5	242	0	17	2
0	29	162	0	5	2	0	23	0	0	0
2	235	2	175	69	0	0	18	0	2	0
48	27	0	0	93	0	0	3	0	1	5
0	33	20	0	6	72	0	19	0	0	0
0	0	24	0	0	0	0	0	0	0	0
0	4	0	0	0	0	40	45	0	2	0
11	17	0	0	26	0	0	4	0	0	4
0	13	0	0	4	0	0	2	0	87	0
36	48	0	0	64	0	0	0	9	0	25

of this problem on the accuracy of our model, we examined two approaches to this problem. In the first approach, we labelled this data as "No Activity." In the second approach, this data was instead removed from the data-set altogether. The resulting models from approach one and approach two were compared, and we found that approach one resulted in a 3-5% improvement in accuracy on average for each model. Moreover, the clusters generated from approach one appeared to be more cohesive. As such, all models discussed in this paper used approach one for the data set. We also tries to implement anomaly detection algorithms which did not turn out great as we realised the data might not be well suited for such a use case. Some of the anomaly detection algorithm that we tried to model were: isolation forest, auto encoders, and DBScan. These models resulted in have an anomaly column that contained either -1 or 1, where -1 denoted an anomaly record and 1 a normal record. Based on the objectives of the projects, we ruled that these results were not helpful in any way.

V. Conclusion

Learning without labels, unsupervised learning, encompasses quite a number of algorithms that have different approaches of learning. It is worth noting that within unsupervised learning, there two major distinctive types of algorithms, namely: clustering algorithms and anomaly detection algorithms. Some of the common clustering algorithms include: KMeans clustering, mean shift clustering, Gaussian mixture and many others. Some of the common anomaly detection algorithms include: DBScan, auto encoders, Isolation forest, and many more. In summary, this report demonstrates usage of number of unsupervised learning algorithms that were implemented on a data set based on activities of daily living data, these algorithms were as follows: K-means clustering, Gaussian Mixture Model, and Means Shift clustering algorithm. Hyper-parameter tuning was also demonstrated on each

of the algorithms to obtain optimal results from these models something which enabled the models achieve descent accuracy scores compared the results before hyper-parameter tuning,

VI. Contributions

All team members equally and actively contributed to this project, in both the code and the report. In terms of individual contributions, each person contributed as follows:

- Justin
  - Code
    - All parts of preprocessing outside of adding of time-based attributes
    - Encoding of dataset
    - Creation, application and hyperparameter tuning of Kmeans model
  - Report
    - Abstract
    - Methodology (Data Preprocessing and Encoding)
- Bailey
  - Code
    - Data visualization
    - Creation, application, and hyperparameter tuning of GMM
  - Report
    - Introduction
    - Methodology (Clustering)
    - Results
- Keith
  - Code
    - Time-based attributes addition to preprocessing
    - Creation, application, and hyperparameter tuning of mean shift
  - Report
    - Methods (Hyper parameter tuning)
    - Discussion
    - Conclusion

References

[1] Bakshi, S., Jagadev, A. K., Dehuri, S., & Wang, G. N. (2014). Enhancing scalability and accuracy of recommendation systems using unsupervised learning and particle swarm optimization. Applied Soft Computing, 15, 21-29.

[2] Shen, B. (2021, January). E-commerce Customer Segmentation via Unsupervised Machine Learning. In The 2nd International Conference on Computing and Data Science (pp. 1-7).

[3] Du Toit, J., Davimes, R., Mohamed, A., Patel, K., & Nye, J. M. (2016). Customer segmentation using unsupervised learning on daily energy load profiles. Journal of Advances in Information Technology Vol, 7(2), 69-75.

[4] Kavakiotis, I., Tsave, O., Salifoglou, A., Maglaveras, N., Vlahavas, I., & Chouvarda, I. (2017). Machine learning and data mining methods in diabetes research. Computational and structural biotechnology journal, 15, 104-116.

[5] Apté, C. (1997). Data mining: an industrial research perspective. IEEE Computational Science and Engineering, 4(2), 6-9.