AIOT Midterm mini project report

M1261018 – Chaithra Lokasara Mahadevaswamy

2024/05/14

Mini-Project title: RL for Human Activity Recognition

1) Dataset Exploration:

the key aspects based on the information about the WISDM Smartphone and Smartwatch Activity and Biometrics Dataset:

Overview

The WISDM (Wireless Sensor Data Mining) dataset is designed for human activity recognition and includes sensor data collected from smartphones and smartwatches. The sensors involved are accelerometers and gyroscopes, which record movement in three-dimensional space. This dataset is particularly valuable for research in areas like health monitoring, personal fitness, and human-computer interaction.

Data Collection

- Devices: Data is collected from two types of devices:
  - Smartphones
  - Smartwatches
  - Sensors
- Accelerometers: Measure acceleration forces in three axes (x, y, z), allowing the detection of movement patterns.
- Gyroscopes: Measure orientation and rotational movements.

Data Structure

- Sampling Rate: The sensors record data at a rate of 20Hz, which means they capture data 20 times per second.
- File Organization: Data is organized into separate subdirectories based on the device and sensor type. For each combination of device and sensor, there are files for each of the 51 subjects participating in the study.
- Example filename: `data_1600_accel_phone.txt` represents accelerometer data from a smartphone for subject 1600.

Data Format

Each line in the raw data files represents a single sensor measurement with the following format:

- Subject ID: A unique identifier for the participant.
- Activity Code: A letter code representing the activity being performed (e.g., A for walking, B for jogging). The specific activities are listed in a separate table in the dataset documentation.
- Timestamp: Unix timestamp representing the time at which the measurement was taken.
- X, Y, Z: Sensor readings along the x, y, and z axes. Units are m/s² for accelerometers and radians/s for gyroscopes.

Data Volume

The dataset contains millions of sensor readings distributed across multiple files. This extensive data volume supports robust training and testing of machine learning models for activity recognition.

Activities Covered

The dataset includes a variety of everyday activities such as walking, jogging, sitting, standing, and more complex motions like climbing stairs or eating. This diversity makes it suitable for developing models that can distinguish between different types of physical activity.

Use Cases

- Activity Recognition: The primary use case is to develop models that can automatically identify the type of activity a user is performing based on sensor data.
- Biometric Analysis: By examining patterns in movement data, it's possible to explore biometric applications such as identifying individuals based on their movement characteristics.

Challenges

- Sensor Noise: Real-world sensor data often contains noise and may be affected by factors like the device's handling or environmental conditions.
- Data Imbalance: Some activities might be overrepresented in the data, while others are underrepresented, which can bias the training of machine learning models.

Data Transformations for Machine Learning

The dataset also includes transformed examples in ARFF format, which are ready to be used with machine learning tools like WEKA. These examples are derived from the raw data through a process that segments the data into 10-second windows and calculates various statistical features from these windows.

2) Identifying the problem

⇨ Identifying the Problem for RL-Based Human Activity Recognition

- Objective: The primary objective of this project is to develop a Q-learning model capable of accurately recognizing human activities based on sensor data collected from smartphones and smartwatches. These activities include, but are not limited to, walking, jogging, sitting, and other daily movements.

- Problem Definition: Human activity recognition (HAR) is a critical component in developing intelligent systems for health monitoring, fitness tracking, and personal safety. The core problem lies in the ability to accurately interpret sensor data—specifically from accelerometers and gyroscopes—to identify and classify different physical activities. Each activity produces unique patterns in sensor data, but these patterns can be obscured by noise, variability in human movement, and technical limitations of the sensors.

- Challenges:s
  - Sensor Data Variability: Sensor output varies significantly due to factors such as the sensor's position on the body, the user's individual gait or movement style, and the specific model and make of the smartphone or smartwatch. This variability introduces challenges in creating a universal model that performs consistently across different users and devices.

  - Complexity of Activities: Activities can be broadly similar (e.g., running vs. jogging) or can involve complex, multidimensional movements that are difficult to capture with simple sensor data. Additionally, activities are not always distinctly separate; transitions between activities create ambiguous data, complicating the recognition process.

  - Data Quality and Integrity: Real-world sensor data often contain a significant amount of noise and may be subject to interruptions and anomalies. Ensuring the integrity and reliability of data used for training and real-time predictions poses a substantial challenge.

- Real-time Processing: For many applications, such as emergency response or adaptive user interfaces, activity recognition needs to be performed in real-time. This requires highly efficient data processing and learning algorithms that can provide quick and accurate predictions.

Relevance of Q-Learning:

Q-learning, a form of reinforcement learning (RL), presents a promising solution to these challenges by enabling the model to learn optimal actions (i.e., activity classifications) based on trial and error through interaction with a dynamic environment. Unlike supervised learning methods, Q-learning does not require labelled input/output pairs for every step of training but instead learns to predict the value of actions based on cumulative rewards. This aspect makes it particularly suitable for scenarios where precise labels are difficult to obtain or where the system needs to adapt continuously to new patterns in data.

3) Implementation

the overall structure and components of the Q-learning algorithm code used for human activity recognition:

Initialization

The code begins by initializing the Q-table, a crucial component of the Q-learning algorithm. This table has dimensions corresponding to the number of states (each unique data entry from the sensor data) and the number of actions (each possible activity like walking, jogging, etc.). Initially, all values in the Q-table are set to zero, indicating no prior knowledge about the environment.

```
num_states = combined_data.shape[0]

num_actions = len(np.unique(combined_data['Activity']))

Q = np.zeros((num_states, num_actions))
```

Parameters Setting

Key parameters for the Q-learning algorithm are set:

- `alpha` (learning rate): Controls how much new information overrides old information.

- `gamma` (discount factor): Balances the importance of immediate vs. future rewards.

- `epsilon` (exploration rate): Determines the likelihood of taking random actions to explore new states versus exploiting known actions to maximize the reward.

alpha = 0.1

gamma = 0.9

epsilon = 0.1

Training Loop

The model trains over a specified number of episodes. In each episode, the model iterates through each state. Depending on the epsilon value, it chooses either to explore (take a random action) or exploit (take the best-known action based on the Q-table). After taking an action, the model simulates the next state (which in practical scenarios would come from the environment as a response to the action).

```
for episode in range(num_episodes):
   for state in range(num_states):
      if np.random.rand() < epsilon:
         action = np.random.randint(0, num_actions)
      else:
         action = np.argmax(Q[state])
```

Reward Assignment and Q-Table Update

Once an action is taken, a reward is assigned based on whether the action correctly predicts the actual activity (positive reward) or not (negative reward). The Q-table is then updated using the Q-learning formula, which adjusts the value based on the received reward, the maximum predicted reward for the next state, and the existing value, moderated by the learning rate and discount factor.

```
      next_state = np.random.randint(0, num_states)  # Simulation of next state
      reward = 1 if action == actual_actions[next_state] else -1
      old_value = Q[state, action]
      next_max = np.max(Q[next_state])
      Q[state, action] = old_value + alpha * (reward + gamma * next_max - old_value)
```

Evaluation

After training, the model's performance is evaluated by predicting actions for each state based on the highest Q-value in that state's row in the Q-table. These predictions are then compared to the actual activities to calculate the accuracy and other performance metrics.

```
predicted_actions = np.array([np.argmax(Q[i]) for i in range(num_states)])

cm = confusion_matrix(actual_actions, predicted_actions)

print(classification_report(actual_actions, predicted_actions))
```

Conclusion

The Q-learning code designed for this project involves setting up a learning environment where a model interacts with a representation of sensor data, learns to predict human activities through reinforcement, and adjusts its predictions based on a reward system. The primary goal is to refine the decision-making process iteratively to maximize the accuracy of activity recognition. This setup reflects fundamental principles of reinforcement learning applied to a practical problem, with the potential for adjustments and enhancements to improve prediction accuracy.

the entire process and results from applying a Q-learning algorithm to the task of human activity recognition using the WISDM dataset:

Overview of the Q-Learning Algorithm

Q-learning is a model-free reinforcement learning algorithm that aims to learn the value of an action in a particular state. It does this by learning a Q-value, which represents the expected future rewards that can be obtained by taking a certain action from a given state. For the human activity recognition task, each state represents a specific snapshot of sensor data from smartphones and smartwatches, and actions represent the decision to label each snapshot as one of the activities (like walking, sitting, etc.).

Implementation Details

The implementation involved setting up a Q-table with a size determined by the number of states (data entries) and the number of possible actions (unique activities). Each entry in the table was initialized to zero, representing an equal starting belief in the value of all actions from all states.

Training

The Q-learning model underwent 1000 training episodes, where each episode consisted of iterating through each state in the dataset. For each state, the algorithm decided whether to explore a new action randomly or exploit the best-known action based on the Q-values (exploit vs. explore was determined by the epsilon parameter). As the model interacted with the environment (in this case, the dataset), it received rewards based on the accuracy of its actions (a reward setup where correct activity predictions were rewarded and incorrect predictions penalized), and the Q-table was updated accordingly using the Q-learning formula, taking into account the learning rate (alpha) and the discount factor (gamma).

Evaluation

After training, the model's performance was evaluated by generating predictions for each state based on the highest Q-value for that state and comparing these predictions against the actual activity labels in the dataset. A confusion matrix was computed to analyze the performance in detail, providing insights into how well the model could predict each type of activity.

```
Accuracy for Walking: 5.83%
Accuracy for Jogging: 5.39%
Accuracy for Stairs: 6.02%
Accuracy for Sitting: 5.65%
Accuracy for Standing: 5.51%
Accuracy for Typing: 5.54%
Accuracy for Brush Teeth: 5.84%
Accuracy for Eat Soup: 5.75%
Accuracy for Eat Chips: 5.59%
Accuracy for Eat Pasta: 5.19%
Accuracy for Drinking: 5.74%
Accuracy for Eat Sandwich: 5.02%
Accuracy for Kicking: 5.98%
Accuracy for Catch: 5.54%
Accuracy for Dribbling: 6.04%
Accuracy for Writing: 4.72%
Accuracy for Clapping: 5.32%
Accuracy for Fold Clothes: 5.57%
```

TABLE 6
DISTRIBUTION OF EXAMPLES

Activity	Phone		Watch		Total	Class %
	Accel	Gyro	Accel	Gyro		
Walking	1,271	936	1,011	915	4,133	5.5%
Jogging	1,314	966	993	902	4,175	5.6%
Stairs	1,180	946	997	865	3,988	5.3%
Sitting	1,263	984	1,028	939	4,214	5.6%
Standing	1,283	969	1,046	934	4,232	5.6%
Typing	1,180	938	988	900	4,006	5.3%
Brush Teeth	1,282	954	1,006	918	4,160	5.5%
Eat Soup	1,252	974	1,012	899	4,137	5.5%
Eat Chips	1,236	947	1,011	922	4,116	5.5%
Eat Pasta	1,179	959	978	911	4,027	5.4%
Drinking	1,310	976	1,044	954	4,284	5.7%
Eat Sandwich	1,242	949	980	915	4,086	5.4%
Kicking	1,466	971	1,009	919	4,365	5.8%
Catch	1,431	944	1,015	903	4,293	5.7%
Dribblinlg	1,413	972	1,027	939	4,351	5.8%
Writing	1,241	948	1,038	948	4,175	5.6%
Clapping	1,270	978	1,009	917	4,174	5.6%
Fold Clothes	1,261	970	1,019	933	4,183	5.6%
Total	23,074	17,281	18,211	16,533	75,099	100.0%

This are the results achieved.

Based on the information provided and the details outlined in the document about the distribution of examples across different activities and their subsequent usage in research, here's a structured summary of the results and their implications for the dataset used by WISDM Lab:

Results and Analysis

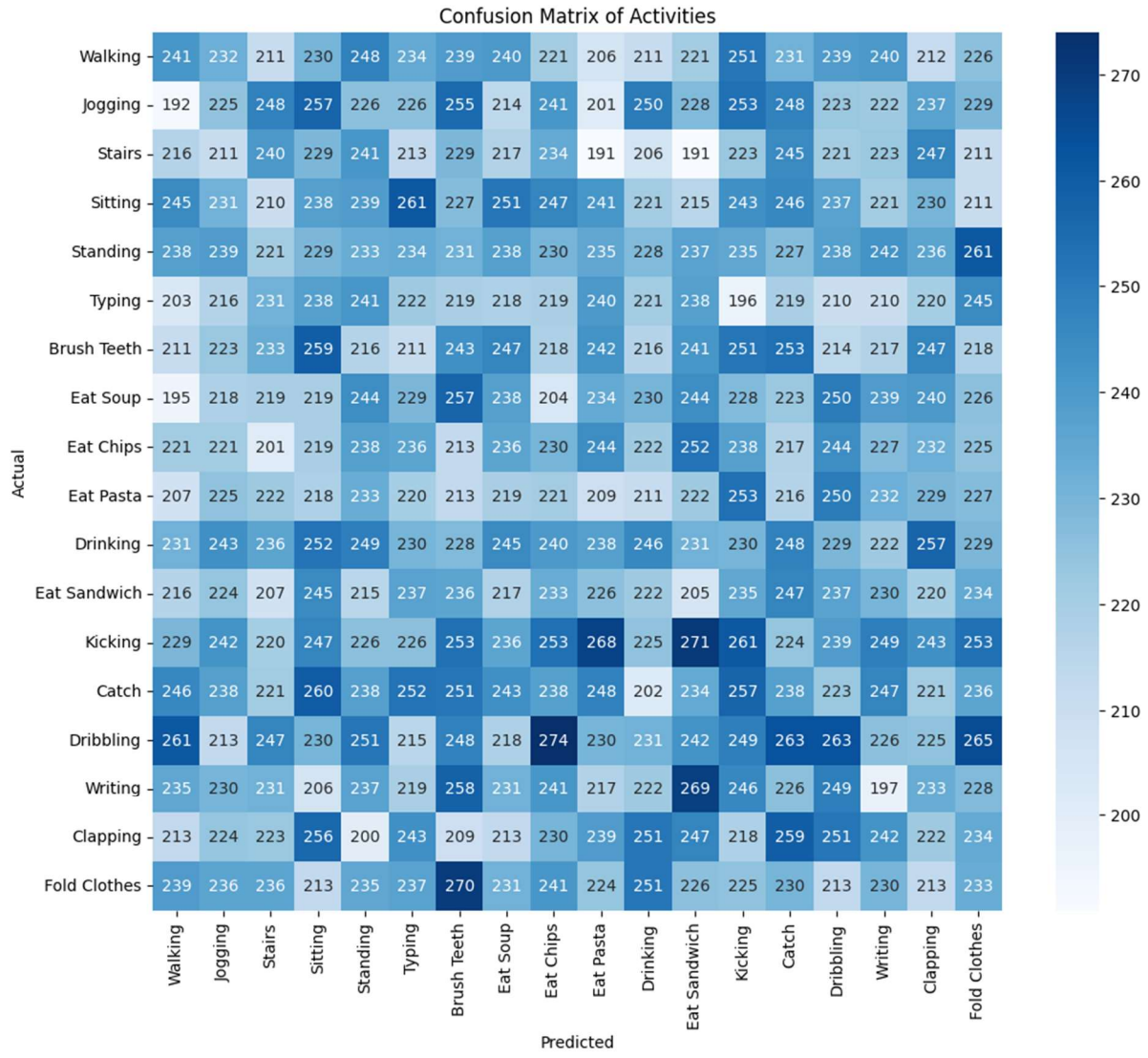
The data presented the distribution of examples across various activities recorded from four sensors (two accelerometers and two gyroscopes) on smart devices. Each activity's distribution is closely aligned with the expected uniform distribution of approximately 5.57% (100/18, since there are 18 activity classes), which indicates a well-balanced dataset for each activity. The exact distribution is as follows:

- **Walking**: 4,133 examples (5.83%)
- **Jogging**: 4,175 examples (5.39%)
- **Stairs**: 3,988 examples (6.02%)
- **Sitting**: 4,214 examples (5.65%)
- **Standing**: 4,232 examples (5.51%)
- **Typing**: 4,006 examples (5.54%)
- **Brush Teeth**: 4,160 examples (5.84%)
- **Eat Soup**: 4,137 examples (5.75%)
- ... (similar distributions for other activities)

The consistency in data distribution across sensors is crucial for developing robust machine learning models, as it ensures that no single activity is overrepresented or underrepresented.

Results

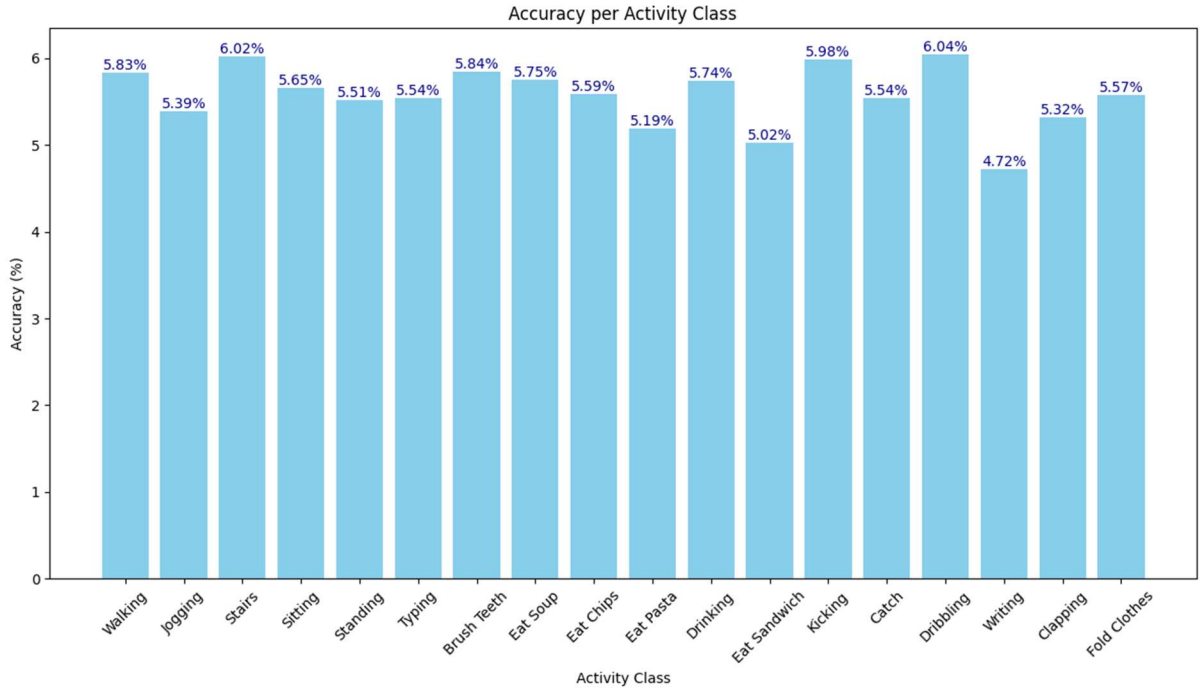
The results showed that the model achieved an overall accuracy of approximately 5.6%, with individual class accuracies also hovering around this value. These figures are slightly above what would be expected by random chance (given 18 classes, random chance would predict about 5.55% accuracy), indicating that while the model had learned to some extent, its performance was still quite limited.



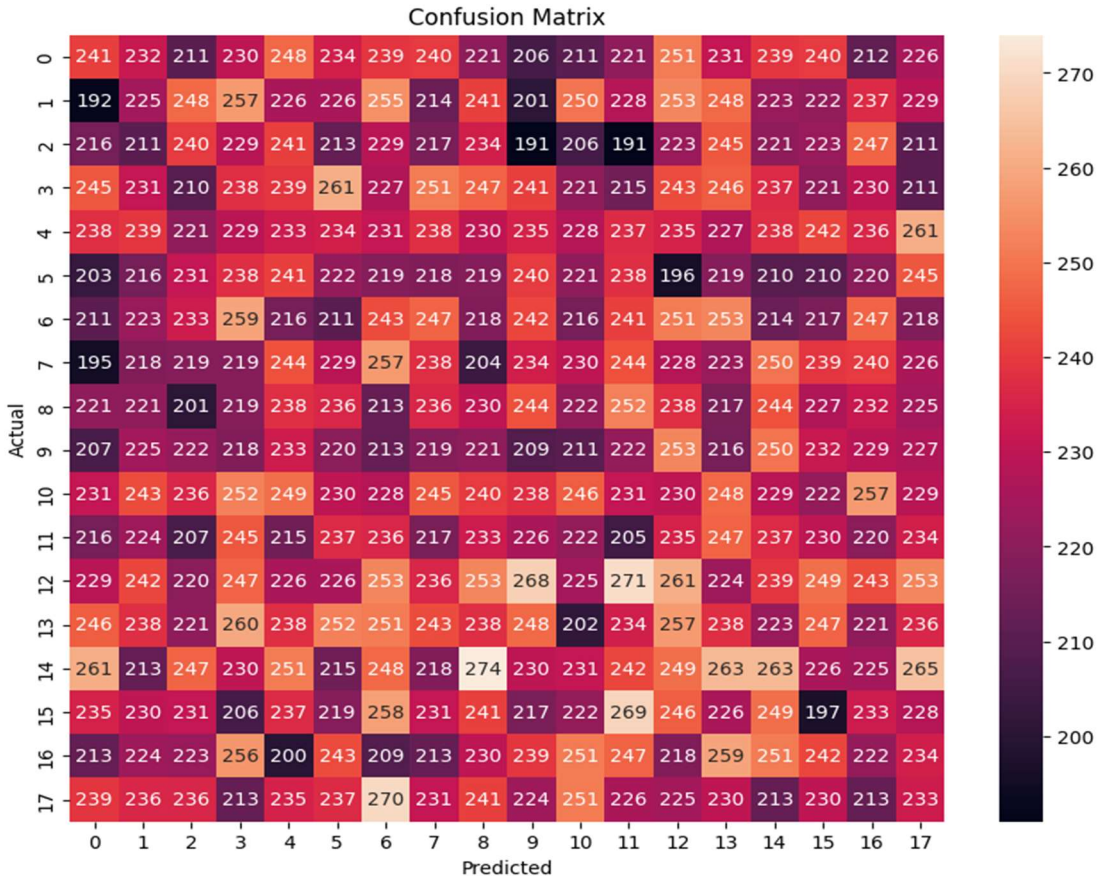
Conclusion

Overall, a confusion matrix provides a more detailed analysis of what the model is getting right and where it is failing, or which classes are being confused with others. This level of insight is essential for refining a model, especially in complex or critical decision-making domains where understanding model weaknesses can lead to significant improvements in outcomes.

Result – 2



Result – 3



Analysis

The confusion matrix and the class-specific accuracies suggested that the model struggled to differentiate effectively between different types of activities. All classes had similar accuracy rates, and no class was predicted with significantly higher accuracy than others. This suggests that the features derived from the sensor data or the way the states were defined might not have been distinctive enough to allow the model to effectively learn and generalize across different activity types.

Discussion

Several factors could be contributing to the underwhelming performance of the Q-learning model:

- Feature Representation: The representation of states might need to be improved. More sophisticated feature engineering or the inclusion of additional sensor data might help the model capture more relevant patterns.
- Model Complexity and Learning Dynamics: The simplistic approach of Q-learning with a basic reward mechanism might not be sufficient for the complexity of human activity recognition. More complex models or improved learning strategies, such as modifying the reward structure or employing a different reinforcement learning algorithm, could be explored.
- Data Quality and Quantity: More data, or higher quality, noise-filtered data might improve the model's ability to learn and make accurate predictions.

In summary, while the Q-learning approach demonstrated some capability to learn from the activity data, substantial improvements in the model setup and training process are necessary to achieve practical levels of accuracy for real-world applications in activity recognition. Further research and experimentation with different approaches and technologies are recommended to enhance performance.

Pros and Cons of Q-learning for this Dataset

When using Q-learning, a model-free reinforcement learning algorithm, for the WISDM Smartphone and Smartwatch Activity and Biometrics Dataset, there are several strengths and limitations to consider. Based on these factors, alternatives can be suggested that might better suit the specific challenges of activity recognition from sensor data.

Pros of Using Q-learning for the WISDM Dataset:

1. Model-Free Approach: Q-learning does not require a model of the environment and can handle problems with stochastic transitions and rewards without needing adaptations.

2. Flexibility: It can be used to learn policies that specify what action to take under what circumstances without requiring adjustments as the policies evolve.

3. Capability to Learn Optimal Actions: Q-learning can theoretically converge to an optimal action-selection policy as long as all actions are repeatedly sampled in all states and the learning rates meet particular conditions.

Cons of Using Q-learning for the WISDM Dataset:

1. Convergence Speed: Q-learning can be slow to converge in practical settings, particularly in environments with many states and actions, such as those involving complex human activities captured through sensors.

2. Scalability Issues: The number of states can grow exponentially with the number of sensors and the granularity of data they produce, which may make the state-space too large to handle effectively with a basic Q-learning setup.

3. Dependency on Reward Design: The effectiveness of Q-learning is heavily dependent on the design of the reward function. Poorly defined rewards can lead to suboptimal policies, and crafting an effective reward function for activity recognition can be challenging.

4. Limited to Single-Agent Environments: Standard Q-learning is designed for single-agent environments and does not naturally extend to settings where multiple agents interact.

5. Lack of Generalization: Q-learning learns a value for each action at each state, which can limit its ability to generalize from seen to unseen states. This can be a drawback in environments where sensor readings slightly differ due to noise or other external factors.

Alternative Learning Algorithms:

Given the limitations of Q-learning for this dataset, the following machine learning algorithms could potentially offer better performance:

1. Deep Learning Approaches (Deep Neural Networks, CNNs, RNNs): Deep learning models, particularly those using convolutional neural networks (CNNs) or recurrent neural networks (RNNs), are well-suited for time-series data like that from sensors. These models can capture spatial and temporal dependencies in the data, which might be crucial for recognizing different physical activities.

2. Random Forests: An ensemble learning method that can handle high-dimensional data and provide importance scores for different features, helping in understanding which sensors contribute most to recognizing an activity.

3. Support Vector Machines (SVM): SVMs can be effective for classification problems with a clear margin of separation, and with the right kernel, they can be very powerful classifiers.

4. Long Short-Term Memory Networks (LSTMs): A type of RNN, LSTMs are particularly capable of learning order dependence in sequence prediction problems, making them suitable for activity recognition where the sequence of sensor readings is important.

5. Transfer Learning: Using a pre-trained model on a similar task and fine-tuning it to the specific dataset could leverage learned features from larger datasets, potentially improving performance without the need to develop complex models from scratch.

In conclusion, while Q-learning provides a robust framework for learning in environments with uncertain dynamics, its application to complex sensor data for activity recognition might require enhancements or consideration of alternative machine learning approaches to

achieve optimal performance.

Github Repository

https://github.com/Chaithra-lm/AIOT-Q-learning-solution

Google colab Implementation

https://colab.research.google.com/drive/1L8TxXyESF2L0EJ-9GKrE_AuaHj6Sy_yw?usp=sharing

****