Ahmedabad University

CSE623 - Machine Learning Theory and Practice
Section - 1

Weekly Report 6
Identify Hard stop and momentary stop using the vehicle trajectory dataset

---

Team Name: **The Overfitters**

Name	Enrolment Number
Jinil Savaj	AU2240159
Jay Raval	AU2240151
Meet Suthar	AU2240198
Karan Prajapati	AU2240161
Vishesh Bhatia	AU2240027

---

1. Objective

The objective of this week's work was to implement, explore, and compare multiple methods for labeling vehicle behaviors into three categories:

- **Moving Vehicle**
- **Momentary Stop**
- **Hard Stop**

We aimed to evaluate different algorithms to determine the most suitable thresholds and logic for robust classification using both rule-based and unsupervised learning techniques.

---

2. Methodology

2.1 Dataset

We used a corrected and preprocessed dataset `df_corrected` containing features such as:

- `speed`, `velocity`, `acceleration`
- `Vxy_smoothed` (smoothed velocity)
- `center_x`, `center_y` (coordinates)
- `jerk` (rate of change of acceleration)

---

3. Implementation

3.1 Labeling Methods

We implemented a Python class `VehicleLabeler`, incorporating five labeling techniques:

3.1.1 Threshold-Based Labeling

A rule-based approach using three speed-related thresholds:

- **Hard Stops**: Detected via sudden drops in speed.
- **Momentary Stops**: Low-speed states not qualifying as hard stops.
- **Moving**: Vehicles above a speed threshold.

Code snippet:

```python
hard_stop = (df['speed'].diff().abs() > hard_stop_thresh) &
(df['speed'] <= momentary_stop_thresh)
```

3.1.2 Time-Based Labeling

This method evaluates the **duration** of stopping behavior:

- Short-duration stops → Momentary
- Long-duration stops (above a time threshold) → Hard stop

Code snippet:

```python
if duration >= min_hard_stop_frames:

    labels[mask] = 2  # Hard stop

else:
```

```
    labels[mask] = 1  # Momentary stop
```

3.1.3 Clustering-Based Labeling (KMeans)

Applies unsupervised learning to identify behavior clusters using:

- `velocity`, `acceleration`, `Vxy_smoothed` Clusters are mapped to labels based on average velocity.

Code snippet:

```python
kmeans = KMeans(n_clusters=3)

clusters = kmeans.fit_predict(X)
```

3.1.4 Hidden Markov Model (HMM)

A statistical model that captures time-dependent transitions between hidden behavioral states using velocity and acceleration.

Code snippet:

```python
model = hmm.GaussianHMM(n_components=3)

states = model.predict(observations)
```

3.1.5 Change Point Detection

Uses the `ruptures` library to detect change points in smoothed velocity time series. Each segment's mean velocity is used to label the behavior.

Code snippet:

```python
algo = KernelCPD(kernel="rbf").fit(speed)

bkps = algo.predict(n_bkps=n_bkps)
```

3.2 Visualization

Each labeling method was visualized using:

- **Spatial distribution** of labeled points
- **Histogram of velocity** across classes
- **Class distribution** bar charts

---

4. DBSCAN Clustering (Unsupervised Analysis)

4.1 Approach

We employed **DBSCAN** to identify behavior patterns without a predefined number of clusters.

Steps:

1. Selected features: `x`, `y`, `Vxy_smoothed`, `acceleration`, `jerk`
2. Scaled features using `StandardScaler`
3. Used k-distance graph to determine optimal `epsilon`

4. Applied `DBSCAN` with `eps = 0.5`, `min_samples = 5`

Code snippet:

```python
dbscan = DBSCAN(eps=0.5, min_samples=5)

clusters = dbscan.fit_predict(X_scaled)
```

4.2 Visualization

- **2D and 3D scatter plots** showing cluster membership
- Noise points (outliers) highlighted in black
- Cluster-wise summary statistics were computed

---

5. Results and Observations

- Threshold and time-based methods are interpretable and parameter-dependent.
- Clustering (KMeans) and HMM detected hidden structures in the data.
- Change point detection segmented behavioral shifts effectively.
- DBSCAN showed potential for unsupervised pattern recognition and identified noise/outliers.

Each method produced varying label distributions. Visualization confirmed that spatial and speed-based patterns differ across techniques.

---

6. Next Steps

- Evaluate method performance using ground truth labels (if available) or expert annotation.
- Optimize hyperparameters for threshold and DBSCAN methods.
- Explore ensemble/hybrid approaches to combine strengths of multiple techniques.
- Investigate supervised models for real-time behavioral classification.

---

7. Challenges

- Difficulty in choosing the best `epsilon` value for DBSCAN without domain-specific guidance.
- Lack of ground truth labels to evaluate labeling accuracy.
- High sensitivity of some methods (especially threshold-based) to parameter tuning.

---

8. Conclusion

This week marked substantial progress in establishing a robust multi-method framework for vehicle behavior classification. The integration of traditional thresholding with modern clustering and probabilistic models has laid the foundation for more accurate and scalable labeling strategies.
