Analisi di classificatori supervisionati su dati radar e LiDAR nel dataset TruckScenes

Silveri Marco – s330394

Politecnico
di Torino

1. Introduzione

In questo lavoro ci proponiamo di analizzare i dati provenienti dal dataset TruckScenes, un insieme di scene raccolte in contesto urbano e autostradale contenente misurazioni provenienti da differenti tipi di sensori. L'obiettivo principale è confrontare le performance di due classificatori supervisionati, Random Forest e Gradient Boosting, nell'identificazione di oggetti basandosi separatamente sui dati radar e LiDAR. La classificazione avviene separatamente per ciascun tipo di sensore e prevede l'attribuzione, a ogni oggetto, di una delle seguenti categorie: dynamic, static, movable o vulnerable.

Poiché radar e LiDAR presentano caratteristiche differenti ci interessa osservare come queste differenze influenzino l'efficacia dei modelli predittivi. Per svolgere questa analisi sono stati utilizzati come strumenti Python, la libreria di machine learning scikit-learn e in particolare la libreria ufficiale truckenes, che permette un accesso strutturato ai dati.

L'approccio seguito prevede la separazione delle scene in set di addestramento e test, l'addestramento dei modelli sui dati grezzi dei due sensori e infine la valutazione delle prestazioni tramite metriche standard come accuratezza, matrice di confusione, f1-score, recall e supporto.

2. Dataset

Abbiamo utilizzato il dataset di MAN chiamato TruckScenes, sviluppato specificamente per camion autonomi. Il dataset completo comprende più di 747 scene, ciascuna della durata di circa 20 secondi, raccolte in vari ambienti: autostrade, strade di raccordo e terminal logistici. La struttura dei dati prevede riferimenti incrociati tramite ID a ogni livello gerarchico. Ogni scena è composta da una serie di sample campionati a 2 HZ, contenenti dati provenienti da 4 telecamere, 6 sensori LiDAR, 6 radar 4D (con copertura a 360°), 2 IMU e GNSS.

Ogni scena ha al proprio interno una serie di annotazioni che coprono 27 classi di oggetti e 15 attributi, con bounding box 3D.

Gli elementi principali del dataset che abbiamo utilizzato sono:

- *scene*: frammento di 20 secondi di viaggio del camion
- *sample*: "istantanea" di una scena
- *sample data*: insieme di dati provenienti da una scena
- *ego_pose:* indica la posizione e l'orientamento del veicolo principale (ego vehicle) in un determinato istante temporale. Serve per georeferenziare i dati raccolti.
- ego_motion_cabin – Rappresenta il movimento della cabina del veicolo (es. accelerazioni, velocità) al timestamp associato. È risultato molto utile quando è stato necessario trasformare i punti dal sistema di riferimento solidale alla cabina del camion al sistema di riferimento inerziale (sistema "mondo")
- *sensor* – Specifica il tipo di sensore utilizzato (es. radar, LiDAR, camera)
- *calibrated_sensor* – Definisce la configurazione e la calibrazione di un determinato sensore sul veicolo (es. posizione e orientamento del sensore rispetto al veicolo)

- *category* – Rappresenta la categoria dell'oggetto secondo una tassonomia predefinita (es. veicolo, persona, ostacolo statico)
- sample_annotation – Un'annotazione che rappresenta un'istanza di oggetto di interesse (es. veicolo) in un preciso sample temporale, completa di bounding box 3D e altri attributi

Sono disponibili 3 dataset:

- mini: versione ridotta di prova (~9 GB), include solo le prime 11 scene
- trainval: contiene annotazioni per 598 scene. Da usare per l'addestramento e la validazione del modello
- test: usato per la valutazione finale del modello su dati mai visti, non contiene annotazioni pubbliche

A causa dell'elevato peso dei dataset, si è deciso di scaricare e utilizzare esclusivamente il primo blocco di scene del dataset di training, composto da 77 scene per un totale di circa 90 GB.

3. Radar VS LiDAR

Radar e LiDAR sono strumenti simili, ma operano in bande diverse dello spettro elettromagnetico: i radar utilizzano onde radio, mentre i LiDAR impulsi laser. Una distinzione fondamentale tra i due riguarda la risoluzione e la risposta in condizioni ambientali difficili: i LiDAR garantiscono un'elevata precisione spaziale, ma risultano vulnerabili a pioggia e nebbia. I radar, al contrario, sono più resistenti agli agenti atmosferici e offrono anche informazioni sulla velocità relativa degli oggetti, qualità essenziale per prevedere i movimenti in scenari complessi. Differenze emergono anche nella densità dei dati raccolti: ad esempio, nel primo campionamento ("first_sample_token") della prima scena, si osservano i seguenti valori aggregati tra tutti i radar e LiDAR installati sul veicolo:

Punti radar	
avg	606.83
max	706
min	471
Punti LiDAR	
avg	40058.83
max	78461
min	18114

Sono stati necessari alcuni accorgimenti per poter gestire ed elaborare l'elevata mole di dati provenienti dai LiDAR, sia in termini computazionali che, soprattutto, di memoria. Questa differenza è dovuta al fatto che i LiDAR scansionano l'ambiente con impulsi laser ad alta frequenza generando una nuvola di punti molto densa e precisa.

4. Metodologia

Per l'analisi dei dati radar e LiDAR del dataset *TruckScenes* sono stati adottati due classificatori supervisionati tra i più noti: **Random Forest Classifier** (RFC) e **Gradient Boosting Classifier** (GBC). Entrambi basati su alberi decisionali, differiscono nella logica di apprendimento: il primo costruisce molteplici alberi decisionali indipendenti e aggrega le predizioni finali tramite voto di maggioranza (*bagging*). Il secondo, invece, costruisce una sequenza di alberi deboli (spesso a bassa profondità), ciascuno dei quali corregge gli errori del precedente, minimizzando progressivamente l'errore tramite il metodo del gradiente.

Delle 77 scene complessive del dataset considerato è stata adottata una **divisione 80/20**, selezionando 61 scene per il training e 16 per il test. Tale scelta segue una prassi comune nella validazione di modelli di machine learning, permettendo un buon compromesso tra quantità di dati per l'apprendimento e robustezza della valutazione.

In fase di training si è optato per una semplificazione del problema, utilizzando direttamente le bounding box annotate nel dataset come input ai classificatori. Questo **approccio semisupervisionato** consente di concentrarsi sulle differenze strutturali tra i dati radar e LiDAR e su come esse influenzino le prestazioni dei modelli, evitando di introdurre complessità aggiuntive legate alla detection o al tracciamento online. Non si tratta quindi di una pipeline pronta all'uso per applicazioni in tempo reale.

Le sei etichette originali del dataset sono state rimappate in **quattro categorie macro**, allo scopo di ridurre la complessità del task e favorire una classificazione più robusta. La mappatura applicata è la seguente:

vehicle	dynamic
ego_vehicle	dynamic
static_object	static
movable_object	movable
human	vulnerable
animal	vulnerable

Una volta definite le etichette, è stato necessario rappresentare ogni **bounding box** tramite un vettore di **feature numeriche**. Alcune di esse condivise tra radar e LiDAR, altre specifiche per il radar. Le feature comuni includono informazioni geometriche e statistiche come il numero di punti, il volume della box, la densità dei punti, la forma (rapporto di allungamento), l'altezza effettiva dei punti, la dispersione spaziale (deviazione standard su X, Y, Z) e la distribuzione verticale dei punti rispetto al centro della box. Questi descrittori catturano la morfologia spaziale dell'oggetto, utile per distinguere tra categorie come statici o mobili.

Per le sole box radar, sono state aggiunte **feature dinamiche** che sfruttano le misure di velocità relativa e Radar Cross Section (RCS), informazioni non disponibili nei dati LiDAR. Tra queste: la velocità media e la sua variabilità (indicativa del movimento interno alla box), il valore medio e la deviazione standard del RCS (che riflette le proprietà di riflettività del materiale), un flag binario che indica se l'oggetto si sta muovendo, e la percentuale di punti quasi statici. Queste feature sono cruciali per sfruttare appieno il potenziale del radar nel riconoscimento di oggetti dinamici.

Per la **fase di valutazione** sono state adottate le metriche messe a disposizione dalla libreria scikit-learn, in particolare: accuratezza, matrice di confusione, precision, recall e f1-score. L'accuratezza misura la percentuale di classificazioni corrette sul totale, mentre la matrice di confusione consente di visualizzare il numero di predizioni corrette ed errate per ciascuna classe. Le metriche precision (precisione), recall (richiamo) e f1-score offrono una valutazione più articolata per ciascuna categoria, considerando il bilanciamento tra falsi positivi e falsi negativi.

Il calcolo di queste metriche è stato effettuato anche in forma aggregata, fornendo così una visione completa delle performance dei modelli su tutti i sensori e classificatori considerati.

5. Implementazione

A livello di implementazione i passi sono stati i seguenti:

1) **Caricamento** dei dati dai sensori a partire dai corrispondenti file PCD (PointCloudData) presenti nel dataset
2) **Filtraggio** punti LiDAR attraverso cKDTree per gravare il meno possibile sulla memoria senza perdere efficacia

```
# Riduzione punti lidar: da ~80 000 a 5 000 punti ----------
if slice_len == 3 and arr.shape[0] > 10_000: # Lunghezza 3 = lidar
    # Scegliamo 10 000 punti casuali per non saturare la RAM
    idx_10k = np.random.choice(arr.shape[0], 10_000, replace=False)
    arr_10k = arr[idx_10k]

    # Costruiamo un KD-Tree sulle sole coordinate XYZ
    kdtree = cKDTree(arr_10k[:, :3])

    # Contiamo quanti vicini (r < 0.25 m) ha ogni punto
    neigh_count = kdtree.query_ball_point(
        arr_10k[:, :3], r=0.25, return_length=True
    )

    # Teniamo i punti con almeno 3 vicini totali
    dense_mask = neigh_count > 2
    arr_dense = arr_10k[dense_mask]

    # Per garantire sempre 5 000 punti finali
    source = arr_dense if arr_dense.shape[0] >= 5_000 else arr_10k
    idx_5k = np.random.choice(source.shape[0], 5_000, replace=False)
    arr = source[idx_5k]
```

3) **Cambio sistema di riferimento** dei punti. È stato necessario passare dal sistema di riferimento solidale al veicolo al sistema di riferimento inerziale (quello delle bounding box). Per i punti LiDAR è stato adottato un approccio "batch", applicando la trasformazione a tutta la nuvola di punti in un'unica operazione
4) Creazione degli oggetti Box, popolamento tramite assegnazione dei punti contenuti e **estrazione delle feature** numeriche descritte in precedenza

```
features = [
    points_num,
    self.volume,
    points_dens,
    self.elongation_ratio,  # Forma dell'oggetto (lungo/schiacciato)
    height,                 # Dimensione verticale box
    std_xyz[0],             # Dispersione lungo X
    std_xyz[1],             # Dispersione lungo Y
    std_xyz[2],             # Dispersione lungo Z
    rel_height,             # Altezza effettiva dei punti nella box
    perc_below_center       # % punti sotto il centro della box
]
```

Figure 1: Features comuni

```
features += [
    avg_rcs,
    sigma_rcs,
    np.linalg.norm(avg_speed_direction),  # Modulo velocità media
    np.linalg.norm(sigma_vel),            # Variabilità velocità
    int(motion_flag),
    percent_quasi_static                  # % punti quasi fermi
]
```

Figure 2: Features esclusive del radar

5) **Addestramento** classificatori Random Forest e Gradient Boosting su tutti i frame di training

```
rfc = RandomForestClassifier(
    n_estimators=100, max_depth=10, random_state=42, class_weight="balanced")
rfc.fit(X_train, y_train)

gbc = GradientBoostingClassifier(
    n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42
)
gbc.fit(X_train, y_train)
```

Figure 3: random_state=42 seed utile a garantire riproducibilità

6) **Valutazione** classificatori per singola scena testando entrambi i modelli su ciascuna delle 16 scene di test

```
for idx in tqdm(test_scenes, desc="Testing scenes", unit="scene"):
    scene_results = evaluate_scene(
        trucksc,
        idx,
        dir_path,
        (rfc_radar, gbc_radar),
        (rfc_lidar, gbc_lidar),
        test_size=None,  # Nessuna divisione tra frame -> usati tutti per il test
    )

    for sensor, res in scene_results.items():
        aggregate[sensor]["y_true"].extend(res["y_true"])
        aggregate[sensor]["rfc"].extend(res["y_pred_rfc"])
        aggregate[sensor]["gbc"].extend(res["y_pred_gbc"])
```

7) Aggregazione e **presentazione** delle metriche globali per ciascun sensore e classificatore

6. Analisi risultati

Sono stati addestrati i modelli su 61 scene per ciascun sensore, pari a 2 440 frame, con un pre-processing molto veloce (meno di trenta secondi complessivi per caricare e indicizzare i ~2,6 milioni di sample data) e questo ha permesso di ottenere classificatori molto precisi.

```
======
Loading truckscenes tables for version v1.0-trainval…
...
2586264 sample_data
…
Done loading in 18.905 seconds.
======
Reverse indexing ...
Done reverse indexing in 7.1 seconds.
======
```

Figure 4

Nelle prove per singola scena radar e LiDAR riconoscono senza difficoltà le classi dynamic e static, raggiungendo di regola un'accuracy fra il 98 % e il 100 %; soltanto quando compaiono le categorie rare (movable e vulnerable) si osserva una lieve flessione ma l'accuracy rimane comunque sempre molto alta.

La **differenza tra i due modelli** emerge chiaramente solo quando si analizzano le metriche per classe, andando oltre la semplice accuratezza complessiva. Sul dataset radar, il Random Forest si dimostra più equilibrato: pur mantenendo un'accuracy globale del 99%, riesce a garantire un recall soddisfacente anche per la classe *movable* (circa 61%). Al contrario, il Gradient Boosting, sebbene ottenga un'accuracy solo leggermente inferiore (98%), mostra un forte calo nel riconoscimento della stessa classe, con un recall che scende fino al 25%. Questo suggerisce che il modello ottimizza meglio le classi maggioritarie a scapito di quelle meno rappresentate. Sul LiDAR accade l'opposto: il Gradient Boosting sale al 99 % grazie a un riconoscimento quasi completo dei soggetti vulnerable, mentre Random Forest resta al 98 % e mostra qualche incertezza in più proprio su soggetti "vulnerable". In entrambi i sensori, però, la categoria movable rimane l'anello debole, sintomo di un forte sbilanciamento del dataset.

Le **matrici di confusione globali** confermano che gli errori residui sono pochissimi e si concentrano quasi esclusivamente negli scambi tra dynamic e static, le due categorie più rappresentate. I valori fuori diagonale relativi a movable e vulnerable sono invece quasi nulli, non perché il modello le riconosca sempre bene, ma semplicemente perché i campioni disponibili sono troppo pochi per generare errori significativi in termini assoluti.

In conclusione, radar e LiDAR offrono prestazioni quasi sovrapponibili sugli oggetti più comuni, ma ciascuno ha un leggero vantaggio su una diversa classe rara: il radar, grazie alle informazioni in più (velocità e RCS), sembra distinguere meglio i piccoli oggetti in movimento, mentre il LiDAR, sfruttando la ricchezza della nuvola di punti, riconosce con maggior continuità i soggetti vulnerabili, come evidenziato dagli alti F1-score raggiunti in quella classe.

```
==== GLOBAL METRICS RADAR - RANDOM FOREST ====

----- RADAR - RANDOM FOREST -----
Accuracy: 0.99
Classification Report:
              precision    recall  f1-score   support

     dynamic       0.99      1.00      0.99      5005
     movable       0.94      0.61      0.74        28
      static       0.99      0.94      0.97       771
  vulnerable       0.64      1.00      0.78        25

    accuracy                           0.99      5829
   macro avg       0.89      0.89      0.87      5829
weighted avg       0.99      0.99      0.99      5829

Confusion Matrix:
[[4987    0    4   14]
 [  11   17    0    0]
 [  43    1  727    0]
 [   0    0    0   25]]

==== GLOBAL METRICS RADAR - GRADIENT BOOSTING ====

----- RADAR - GRADIENT BOOSTING -----
Accuracy: 0.98
Classification Report:
              precision    recall  f1-score   support

     dynamic       0.99      1.00      0.99      5005
     movable       0.78      0.25      0.38        28
      static       0.98      0.93      0.95       771
  vulnerable       0.64      1.00      0.78        25

    accuracy                           0.98      5829
   macro avg       0.85      0.79      0.78      5829
weighted avg       0.98      0.98      0.98      5829

Confusion Matrix:
[[4987    1    3   14]
 [   8    7   13    0]
 [  55    1  715    0]
 [   0    0    0   25]]
```

```
==== GLOBAL METRICS LIDAR - RANDOM FOREST ====

----- LIDAR - RANDOM FOREST -----
Accuracy: 0.98
Classification Report:
              precision    recall  f1-score   support

     dynamic       0.99      1.00      0.99      4655
     movable       0.56      0.39      0.46        38
      static       0.99      0.97      0.98      1473
  vulnerable       0.74      0.89      0.81        36

    accuracy                           0.98      6202
   macro avg       0.82      0.81      0.81      6202
weighted avg       0.98      0.98      0.98      6202

Confusion Matrix:
[[4635    5    4   11]
 [  20   15    3    0]
 [  47    4 1422    0]
 [   0    3    1   32]]

==== GLOBAL METRICS LIDAR - GRADIENT BOOSTING ====

----- LIDAR - GRADIENT BOOSTING -----
Accuracy: 0.99
Classification Report:
              precision    recall  f1-score   support

     dynamic       0.99      1.00      0.99      4655
     movable       0.58      0.39      0.47        38
      static       0.99      0.96      0.98      1473
  vulnerable       1.00      0.94      0.97        36

    accuracy                           0.99      6202
   macro avg       0.89      0.82      0.85      6202
weighted avg       0.98      0.99      0.99      6202

Confusion Matrix:
[[4649    2    4    0]
 [  14   15    9    0]
 [  51    7 1415    0]
 [   0    2    0   34]]
```

7. Conclusione

Possiamo quindi sostenere che il sistema analizzato si è dimostrato valido come primo livello di riconoscimento soprattutto per oggetti statici e dinamici in scenari relativamente controllati. Tuttavia, non può essere considerato sufficiente in contesti complessi o critici, dove la presenza di oggetti rimovibili o entità vulnerabili (come esseri umani o animali) richiede una maggiore affidabilità e sensibilità nella classificazione. La metodologia adottata presenta infatti almeno le seguenti limitazioni:

- un numero maggiore di scene testate permetterebbe una generalizzazione più affidabile dei risultati ottenuti
- l'esclusione di approcci basati sul deep learning riduce significativamente il potenziale di apprendimento automatico, soprattutto in contesti più complessi o variabili

Per superare tali limiti e rendere il sistema applicabile a scenari reali sarebbe utile considerare almeno l'integrazione di tecniche di sensor fusion, che combinino dati provenienti da sensori eterogenei per aumentare robustezza e ridondanza informativa.

In definitiva, il sistema costituisce una buona base sperimentale, ma richiede ulteriori sviluppi per poter essere impiegato in ambienti dinamici reali e in applicazioni critiche per la sicurezza dei passeggeri all'interno del veicolo e dei soggetti vulnerabili intorno ad esso.

8. Integrazione pipeline

Infine, ritengo opportuno segnalare il lavoro svolto da Emanuele Sanna, il quale si configura come complementare al mio. In particolare, il suo obiettivo è quello di generare automaticamente le bounding box a partire dai dati grezzi forniti dai sensori.

Emerge una possibile integrazione tra i due lavori. In particolare, il suo sistema potrebbe essere impiegato a monte del mio, come fase preliminare di generazione automatica delle regioni di interesse, le quali verrebbero poi analizzate e classificate dal mio modello. Una tale pipeline modulare permetterebbe di ottenere un sistema autonomo, riducendo la dipendenza da annotazioni manuali.

Link utili:

- https://brandportal.man/d/QSf8mPdU5Hgj
- https://brandportal.man/d/QSf8mPdU5Hgj/devkit-tutorial#/-/dataset-schema
- https://github.com/TUMFTM/truckscenes-devkit?tab=readme-ov-file