UNIVERSITÀ DI CATANIA

Dipartimento di Matematica e Informatica

Laurea Triennale in Informatica

Kevin Speranza

[TITOLO PROGETTO]

Big Data Project

Professore: Alfredo Pulvirenti

Academic Year 2024 - 2025

Contents

Chapter 1

Introduction

Chapter 2

Dataset

2.1 MovieLens Dataset

Per questo progetto abbiamo utilizzato il dataset **MovieLens**, una delle fonti più comuni e ben strutturate per task di raccomandazione. Il dataset contiene informazioni su:

- **Utenti**: identificati da un ID univoco (nessuna informazione demografica è stata utilizzata).
- **Film**: ciascun film ha un ID, un titolo e un elenco di *generi* associati.
- **Rating**: ogni interazione tra utente e film è rappresentata da un voto (valori tra 0.5 e 5.0), fornito da un utente per un determinato film.

2.1.1 Costruzione della Rete Bipartita

Come mostrato nella figura 2.1 la rete è stata costruita come una **rete bipartita**, ovvero un grafo composto da due insiemi distinti di nodi, nei quali gli archi possono collegare solo nodi appartenenti a insiemi diversi. In questo caso:

- Un insieme di nodi rappresenta gli *utenti*.
- L'altro insieme di nodi rappresenta i *film*.
- Gli archi collegano esclusivamente utenti e film, indicando un'interazione sotto forma di rating assegnato dall'utente a quel film.
- Ogni arco è quindi **pesato** con il valore del rating corrispondente, rappresentando così l'intensità o preferenza dell'utente per quel film.

La struttura bipartita è fondamentale per applicare GraphSAGE in modo efficace, trattando utenti e film come classi distinte ma connesse tramite i loro comportamenti.

2.1.2 Engineering degli Attributi dei Nodi

Ogni nodo nella rete è arricchito con un vettore di attributi che cattura le sue caratteristiche principali.

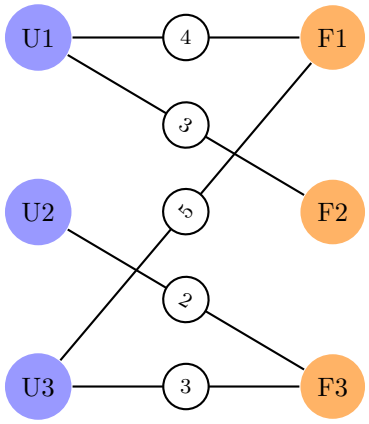

Figure 2.1: Esempio rete bipartita tra utenti (blu) e film (arancione), con pesi sugli archi che rappresentano i rating degli utenti.

Vettore Film

Per ciascun film abbiamo calcolato:

- **film_id**: un identificativo univoco per ciascun film.
- **vettore dei generi** (genre_x): un vettore binario di lunghezza N ottenuto tramite **one-hot encoding** dei generi disponibili, ovvero ogni posizione indica l'appartenenza o meno del film a un genere (ad esempio: *[1, 0, 1, 0, ...]*). Successivamente, questo vettore viene normalizzato per tenere conto della distribuzione complessiva dei generi.
- **median**: il **rating mediano** ricevuto dal film, calcolato come mediana di tutti i rating forniti dagli utenti.

film_id	genre_unknown	genre_...	genre_Western	**median**
1	0.0450	...	0.0150	4.0

Table 2.1: Esempio di vettore per un film.

Vettore Utenti

Per ciascun utente abbiamo calcolato:

- **user_id**: un identificativo univoco per ciascun utente.
- **vettore aggregato dei generi** (genre_x): ottenuto sommando i vettori one-hot normalizzati dei film recensiti dall'utente e poi normalizzando il risultato.
- **median**: il **rating mediano** assegnato dall'utente, calcolato considerando tutti i voti che ha dato.

user_id	genre_unknown	genre_...	genre_Western	**median**
1	0.0450	...	0.0150	4.0

Table 2.2: Esempio di vettore per un utente.

Chapter 3

Implementazione

partendo dal grafo bipartito definito abbiamo bisogno di creare embedding al fine di poter effettuare operazioni / raccomandazioni. a tal scopo è stato scelto l'uso di graphsage.

[parla degli aspetti teorici]

3.1 GraphSAGE

GraphSAGE (*Graph Sample and Aggregate*) è un algoritmo di convoluzione sui grafi che costruisce rappresentazioni (embedding) dei nodi basandosi su un meccanismo iterativo di aggregazione delle informazioni dai nodi vicini.

Il processo generale per un nodo v al layer k è:

$$h_v^{(k)} = \sigma\left(W^{(k)} \cdot \text{AGGREGATE}^{(k)}\left(\{h_u^{(k-1)} \mid u \in \mathcal{N}(v)\} \cup \{h_v^{(k-1)}\}\right) + B^{(k)}\right)$$

dove:

- $\mathcal{N}(v)$ è l'insieme dei vicini di v,
- $h^{(k)}$ è l'embedding al layer k,
- $\text{AGGREGATE}^{(k)}$ è la funzione di aggregazione scelta,
- σ è una funzione di attivazione (ReLU nel nostro caso),
- $W^{(k)}, B^{(k)}$ sono i parametri del layer.

3.2 Architettura del Modello

Il modello è stato implementato in **PyTorch Geometric**. La struttura della rete è flessibile rispetto al numero di layer e alla dimensione degli embedding.

[scrivi discoriva questsa parte]:

- Il numero di layer è variabile (da 1 a 3).
- Nei layer intermedi viene applicata `ReLU` seguita da `Dropout`.
- L'ultimo layer restituisce direttamente gli embedding finali.

3.3 Sperimentazione con i Livelli di Convoluzione

Per valutare le prestazioni del modello, abbiamo variato:

- **Numero di layer**: 1, 2, 3
- **Dimensione degli embedding**: 32 e 64 e 128

Tutte le configurazioni sono state addestrate per 500 epoche usando l'ottimizzatore Adam con learning rate $\alpha = 10^{-3}$.

—

3.4 Generazione delle Raccomandazioni

Una volta ottenuti gli embedding finali h_v per ogni nodo:

1. Per ogni utente u, si calcolano le distanze tra il suo embedding e quelli di tutti i film.
2. Vengono selezionati i k film più vicini (ad esempio $k = 10$) come raccomandazioni.

Come distanza abbiamo sperimentato sia la *cosine similarity* che la distanza euclidea.

3.5 Valutazioni

Per valutare la qualità delle raccomandazioni, sono state utilizzate le seguenti metriche:

- **Precision@K** e **Recall@K**: misura la qualità dei top-K suggeriti rispetto alle preferenze effettive.
- **Mean Average Precision (MAP)**: media delle precisioni ai diversi cut-off.
- **AUC-ROC**: tratta la raccomandazione come classificazione binaria.

I risultati mostrano che due layer con embedding da 64 offrono un buon compromesso tra accuratezza e complessità computazionale.

Chapter 4

Results

Conclusion

Bibliography