

UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTÀ DI SCIENZE E TECNOLOGIE



Corso di Laurea magistrale in
Informatica

IL TITOLO DELLA TESI

Relatore: Relatore 1
Correlatore: Correlatore 1

Tesi di Laurea di:
Lorenzo D'Alessandro
Matr. Nr. 939416

ANNO ACCADEMICO 2020-2021

Dedica

Ringraziamenti

Questa sezione, facoltativa, contiene i ringraziamenti.

Indice

| | |
|---|------------|
| Ringraziamenti | ii |
| Indice | iii |
| 1 Introduzione | 1 |
| 1.1 I contenuti | 1 |
| 1.2 Organizzazione della tesi | 1 |
| 2 Stato dell'arte | 2 |
| 3 RS mobile | 3 |
| 3.1 Introduzione | 3 |
| 3.2 Limitazioni dei RS tradizionali | 4 |
| 3.3 Nome modello | 6 |
| 3.3.1 Struttura modello | 7 |
| 4 Capitolo 4 | 9 |
| 5 Capitolo 5 | 10 |
| 6 Conclusioni | 11 |
| 6.1 Conclusioni | 11 |
| 6.2 Sviluppi futuri | 11 |
| Bibliografia | 12 |

Capitolo 1

Introduzione

Introduzione...

1.1 I contenuti

Spiegazione problema...

1.2 Organizzazione della tesi

Organizzazione tesi...

Capitolo 2

Stato dell'arte

Capitolo 3

RS mobile

3.1 Introduzione

Nelle reti opportunistiche, di solito, un dispositivo mobile dovrebbe essere in grado di condividere informazioni con altri dispositivi in prossimità in modo da scoprire contenuti utili per il suo proprietario. In questo capitolo è descritto un nuovo tipo di sistema di raccomandazione adatto a questo scenario, in grado di suggerire i contenuti più adatti per un utente tra quelli scoperti nelle vicinanze. Infatti, un problema è che non è semplice estendere le soluzioni esistenti per adattarle alla limitazioni di questo nuovo scenario. I recommender system tradizionali si appoggiano su un modello client/server centralizzato, in cui il sistema di raccomandazione esegue sul server, e processa le richieste in arrivo dai client che possono essere dispositivi mobili o fissi. Oltre a questo, per le reti opportunistiche è molto diverso anche il task di raccomandazione. Tipicamente un RS deve imparare a prevedere i rating sulle coppie utente-oggetto per poter riempire i valori mancanti nella matrice dei rating. Questa matrice può essere vista come la conoscenza globale del sistema su tutti gli utenti e gli oggetti disponibili, e sulle valutazioni che gli utenti hanno dato agli oggetti. Nel caso dei RS collaborative-filtering il recommendation engine impara un modello singolo che sarà usato per fare raccomandazione su tutti gli utenti del sistema, mentre nel caso dei RS content-based sarà istanziato un modello per ogni utente che comunque ha una conoscenza globale di tutti gli oggetti del sistema. Al contrario nelle reti opportunistiche ogni dispositivo potrebbe essere a conoscenza solo di una piccola parte dell'informazione globale. Questa conoscenza globale è inizialmente correlata alle sole informazioni sull'utente locale, poi cresce con lo scambio di informazioni tramite comunicazione D2D con altri utenti o dispositivi di altra natura. Quindi, nelle reti opportunistiche ogni dispositivo impara un modello di raccomandazione sfruttando solo le informazioni raccolte dal suo utente locale.

Un'altra caratteristica importante per un sistema di raccomandazione per dispositivi mobili è il supporto per il contesto utente. Sfruttando i numerosi sensori presenti sui dispositivi degli utenti è possibile generare un contesto ad alta dimensionalità che caratterizza in modo migliore la situazione dell'utente e permette raccomandazioni più accurate. Al contesto fisico si può aggiungere il contesto sociale, che descrive ad esempio che tipo di persone l'utente ha intorno a sé (amici, colleghi, sconosciuti) e con che tipo di persone ha scambiato le proprie informazioni, e quindi da chi ha ricevuto una raccomandazione.

3.2 Limitazioni dei RS tradizionali

Gli algoritmi di raccomandazione stato dell'arte descritti nel Capitolo 2 hanno alcune limitazioni che rendono impossibile un'implementazione senza modifiche in ambiente mobile.

ALS

Alternating least square (descritto in ...) è un algoritmo di matrix factorization per feedback impliciti. L'input di ALS è una matrice R di dimensione $u \times i$ con u numero di utenti, e i numero di oggetti. Un elemento r_{ui} della matrice R indica quante volte l'utente u ha consumato l'oggetto i . In ambiente mobile inizialmente la matrice R è vuota perché l'utente non ha espresso nessuna valutazione e non ha incontrato altri utenti. Ogni volta che un nuovo utente o un nuovo oggetto viene scoperto si aggiunge una nuova riga / colonna. Il primo problema di questo algoritmo, e più in generale degli algoritmi di matrix factorization è quindi che l'aggiunta di un nuovo utente/oggetto comporta un cambiamento della dimensione della matrice R e l'algoritmo deve essere addestrato di nuovo per poter raccomandare i nuovi oggetti. Inoltre il tempo di training cresce linearmente con il numero di utenti e oggetti [Verificare]. Il secondo problema di MF è l'aggiunta del contesto. Come descritto in [sezione context-aware] ogni feature di contesto è una dimensione aggiunta alla matrice dei rating R . Con l'aumento delle dimensioni, lo spazio dimensionale aumenta in modo esponenziale [1]. Questa crescita esponenziale porta un'alta sparsità dei dati, fino a quando ogni punto è equidistante dagli altri. La ??, mostra uno spazio dimensionale diviso in 4 regioni. Aggiungendo una dimensione lo spazio cresce esponenzialmente a 16 regioni. Aggiungendo una terza dimensione si arriva a 64 regioni.

NeuMF

Neural matrix factorization (descritto...) implementa un RS collaborative-filtering con una deep neural network. L'input della rete è del tipo user-id, item-id, rating. Per essere compresi dalla rete user-id e item-id sono convertiti con one-hot encoding

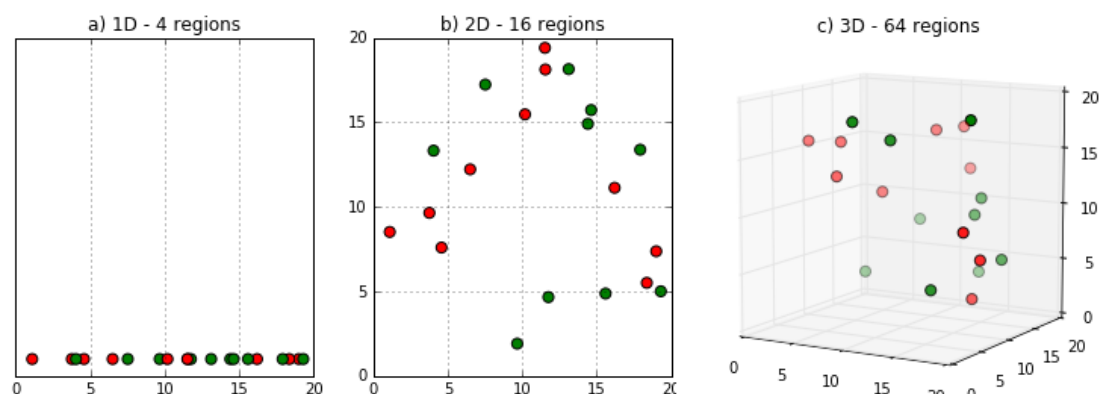


Figura 1: Curse of dimensionality

in vettori binari. Questi vettori hanno lunghezza pari al numero di user/item e hanno valore 1 solo in corrispondenza della cella numero user-id, il resto sono 0. La dimensione dei vettori di embedding è quindi da definire a livello di compilazione della rete prima di eseguire il training. Una volta terminato il training, passati alla fase di utilizzo, l'input della rete deve avere la stessa dimensione impostata a livello di compilazione. Come per ALS quindi aggiungere un nuovo utente od un nuovo oggetto significa dover compilare di nuovo il modello ed eseguire di nuovo il training da zero. Questo modello inoltre, come presentato nel paper originale [2] non ha nessun supporto per le feature di contesto.

Context-aware NeuMF

Context-aware neural matrix factorization (descritto ...) è un'estensione di NeuMF che supporta le feature di contesto. Questo modello come dimostrato nel paper in cui è stato proposto, non soffre del problema di curse of dimensionality di matrix factorization. È quindi possibile dare in input alla rete un vettore del tipo [user-id, item-id, context, rating], in cui il vettore context può avere un'alta dimensionalità. Risolve quindi il problema dell'integrare le numerose feature di contesto fisico estratte dai sensori degli smartphone e le features sociali che descrivono il contesto sociale dell'utente. Rimangono le stesse limitazioni di NeuMF sugli utenti e gli oggetti che sono sempre dati in input come vettori in one-hot encoding definiti a livello di compilazione.

3.3 Nome modello

I problemi principali nell'implementare un sistema di raccomandazione per sistemi mobili e pervasivi quindi sono principalmente due:

1. Non si conosce a priori il numero di utenti e oggetti presenti nel sistema perchè l'utente ne scopre di nuovi gradualmente tramite interazione D2D.
2. È difficile integrare le numerose informazioni di contesto fisico generate dai dispositivi mobili e le informazioni di contesto sociale che descrivono la situazione dell'utente.

In questa sezione viene descritta l'implementazione di un recommender system context-aware basato su deep-learning e pensato per dispositivi mobili con le seguenti caratteristiche:

1. L'algoritmo di raccomandazione esegue la fase di training e inferenza direttamente su dispositivo mobile.
2. Non c'è un architettura client-server con un server che ha conoscenza globale dell'informazioni.
3. Ogni utente ha una propria istanza locale dell'algoritmo di raccomandazione
4. Le valutazioni sono feedback impliciti con valore 0 o 1.
5. L'algoritmo di raccomandazione utilizza sia i feedback dell'utente locale che quelli ricevuti da altri utenti.
6. Il recommender system supporta informazioni di contesto fisico e sociale.

Normalmente l'input di un modello collaborative filtering è composto da tuple (user-ID, item-ID, physical context). Si può sostituire item-ID con delle feature che descrivono l'oggetto, esattamente nello stesso modo in cui operano i sistemi di raccomandazione context aware. Ad esempio se si sta sviluppando un RS per raccomandazione per consigliare ristoranti agli utenti, si può sostituire l>ID del ristorante con feature specifiche come il tipo di cibo servito, il suo prezzo, l'atmosfera, se ha sedute all'aperto, etc. Allo stesso modo si può sostituire lo user-ID con delle feature che descrivono l'utente. Queste possono essere feature generiche che descrivono l'utente come l'età e il sesso, a cui si aggiungono feature specifiche per il RS che si sta sviluppando. Tornando all'esempio dei ristoranti si potrebbe chiedere all'utente quanto è disposto a spendere per mangiare fuori e il tipo di cucina preferita. A feature di utente e oggetto si aggiungono le feature del contesto fisico e sociale. Un'istanza di rating per il modello proposto in questa tesi è quindi del tipo [user-features, item-features, physical context, social context].

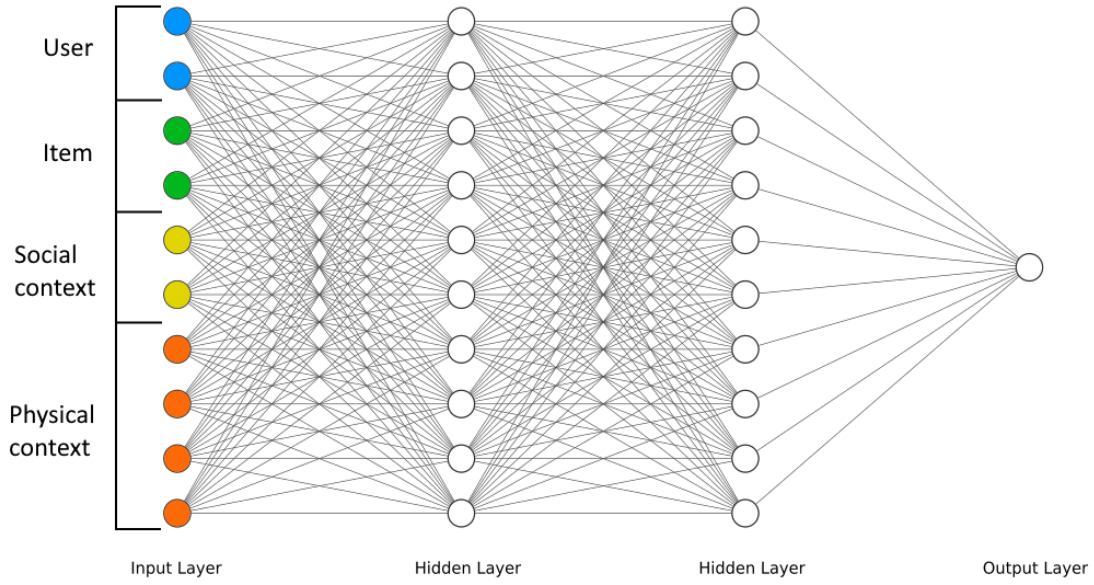


Figura 2: Schema di nome modello

3.3.1 Struttura modello

L'istanza di ratings descritta prima è data in input ad una rete feed-forward fully connected. Una rete feed-forward non contiene cicli nel suo grafo [3], fully connected indica che ogni neurone del layer i è connesso a tutti i neuroni del layer $i + 1$. La rete ha un layer di input, un layer di output e n layer nascosti. Il layer di input ha un numero di neuroni pari alle feature in ingresso (sommando user, item e context feature), il layer di output ha sempre un neurone, mentre il numero di neuroni nei layer nascosti, e il numero di layer nascosti si può trovare tramite grid search o random search. In questa tesi è stato utilizzato lo stesso numero di neuroni in ogni layer nascosto, ma si può ad esempio adottare un design a torre in cui i layer più profondi hanno meno neuroni di quelli dei primi layer. Come funzione di attivazione del layer di input e dei layer nascosti ho scelto la funzione rectified linear unit (ReLU) definita come $f(x) = \max\{0, x\}$. La funzione ReLU è consigliata per la maggior parte delle reti feed-forward [3] e ha diversi vantaggi rispetto a funzioni di attivazione come sigmoide e tanh: è più plausibile biologicamente, non viene saturata (a differenza della sigmoide che ha al massimo valore 1), e incoraggiando l'attivazione sparsa dei neuroni rende più difficile che si verifichi l'overfitting del modello durante il training [4]. Come funzione di attivazione del layer di output ho scelto la funzione sigmoide definita come

$$f(x) = \frac{1}{1 + e^{-x}}$$

che restringe l'output della rete in valori tra 0 e 1, ed è quindi adatta per problemi di classificazione binaria [5]. Come funzione di loss la scelta classica per un classificatore binario è la binary cross-entropy / log loss, definita come

$$H_p(q) = -\frac{1}{N} \sum_{i=1}^N y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i))$$

Capitolo 4

Capitolo 4

Capitolo 5

Capitolo 5

Capitolo 6

Conclusioni

6.1 Conclusioni

Conclusioni...

6.2 Sviluppi futuri

Sviluppi futuri...

Bibliografia

- [1] Prasad Pore. What is the curse of dimensionality? <https://www.kdnuggets.com/2017/04/must-know-curse-dimensionality.html>, 2017.
- [2] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web, WWW '17*, page 173–182, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.
- [3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.
- [4] Xavier Glorot, Antoine Bordes, and Y. Bengio. Deep sparse rectifier neural networks. volume 15, 01 2010.
- [5] Jason Brownlee. How to choose an activation function for deep learning. <https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/>, 2021.