

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 Un RS per dispositivi mobili e pervasivi	3
3.1 Introduzione	3
3.2 Limitazioni dei RS tradizionali	4
3.3 Nome modello	6
3.3.1 Struttura modello	7
3.4 Preprocessing dei dati	8
3.4.1 Contesto sociale	8
4 Capitolo 4	11
5 Capitolo 5	12
6 Conclusioni	13
6.1 Conclusioni	13
6.2 Sviluppi futuri	13
Bibliografia	14

Capitolo 1

Introduzione

Introduzione...

1.1 I contenuti

Spiegazione problema...

1.2 Organizzazione della tesi

Organizzazione tesi...

Capitolo 2

Stato dell'arte

Capitolo 3

Un RS per dispositivi mobili e pervasivi

3.1 Introduzione

In questo capitolo viene descritta l'implementazione di un algoritmo di raccomandazione pensato per dispositivi mobili e pervasivi immersi in un ambiente totalmente distribuito come quello delle reti opportunistiche. In queste reti, 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. Un sistema di raccomandazione può essere utile per filtrare i contenuti più adatti ad un utente tra quelli scoperti nelle vicinanze. Non è però semplice estendere le soluzioni esistenti per adattarle alle limitazioni imposte da 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, il task di raccomandazione per un RS pervasivo è diverso rispetto a quello di un RS client/server. 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 è inizialmente circoscritta alle sole informazioni sull'utente locale, per poi allargarsi con lo

scambio di informazioni tramite comunicazione device-to-device (D2D) con altri utenti o dispositivi di altra natura. Di conseguenza, un RS implementato in un ambiente distribuito impara un modello di raccomandazione basato unicamente sulle 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 accurato la situazione dell'utente e permette raccomandazioni più accurate. Al contesto fisico estratto dai sensori o altre fonti, si può aggiungere il contesto sociale che descrive quali tipologie di persone l'utente ha intorno a sé (amici, colleghi, sconosciuti, etc.) 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 difficile l'implementazione senza alcuna modifica nell'ambiente di riferimento.

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 alla matrice R . Il primo problema di questo algoritmo, e più in generale degli algoritmi di matrix factorization è l'aggiunta di un nuovo utente/oggetto che comporta un cambiamento nella dimensione della matrice R . L'algoritmo deve essere addestrato nuovamente da zero sulla nuova matrice R per poter raccomandare i nuovi oggetti e tenere conto delle valutazioni dei nuovi utenti. Il secondo problema di MF è l'aggiunta del contesto. Come detto 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] mentre il numero dei punti nello spazio rimane sempre lo stesso. Questa crescita esponenziale porta un'alta sparsità dei dati, che impedisce al modello di generalizzare correttamente. La Figura 1, mostra graficamente il problema. Nell'immagine a) lo spazio 1D è diviso in 4 regioni. In b) aggiungendo una dimensione lo spazio cresce esponenzialmente a

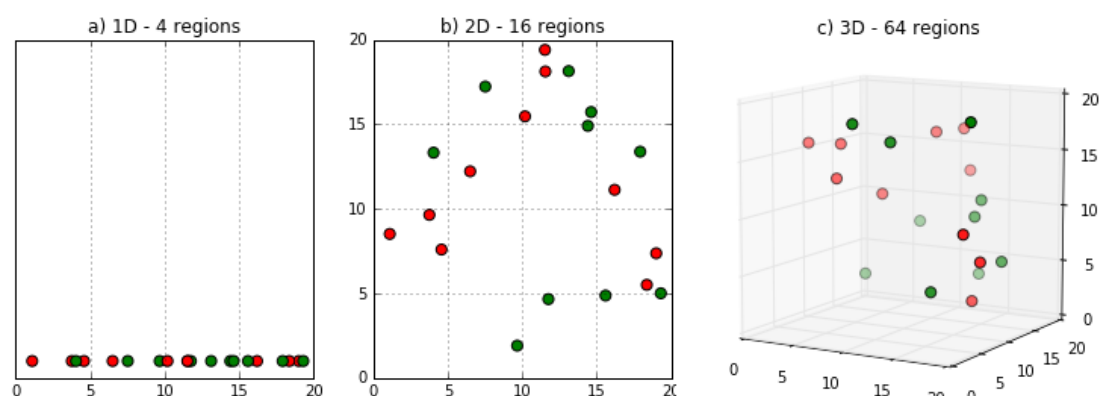


Figura 1: Curse of dimensionality

16 regioni. In c) aggiungendo una terza dimensione lo spazio è diviso in 64 regioni ma il numero di punti in ogni regione rispetto ad a) è molto minore.

NeuMF

Neural matrix factorization (descritto...) implementa un RS collaborative-filtering con una deep neural network. L'input della rete è un record formato da `[user_id, item_id, rating]`. Per essere correttamente processati dalla rete neurale `user_id` e `item_id` sono convertiti da numeri interi in vettori binari con one-hot encoding. Questi vettori hanno lunghezza pari al numero di user/item e hanno valore 1 solo in corrispondenza dell'elemento numero `user_id/item_id`, il resto degli elementi ha valore 0. La dimensione dell'input della rete e di conseguenza la dimensione dell'input del layer di embedding è da definire a livello di compilazione della rete prima di eseguire il training. Una volta terminato il training, la rete è in grado di predire il rating degli stessi utenti/oggetti già presenti durante il training. Questo perché l'input deve avere la stessa dimensione di quello definito a livello di compilazione, e quindi il numero di utenti e oggetti è lo stesso definito 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 essendo basato su reti neurali, è meno soggetto di matrix factorization al problema della curse of dimensionality. L'input della rete è un record formato da `[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 sull'input, dato che gli ID di utenti e oggetti sono dati in input come vettori in one-hot encoding con dimensione fissata.

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

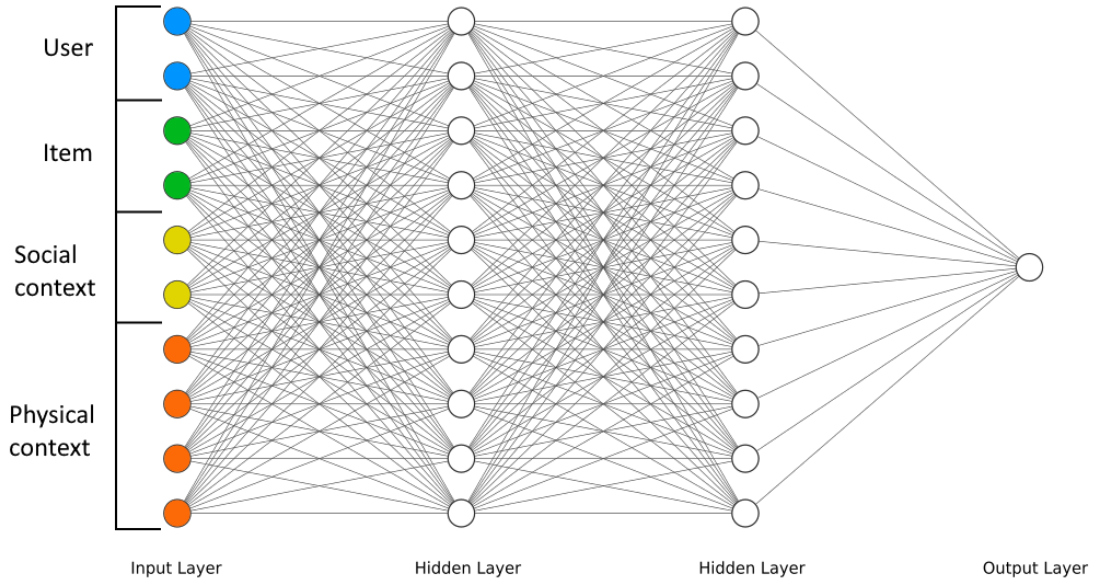


Figura 2: Schema di nome modello

delle feature che descrivono l'utente. Queste possono essere feature generiche che descrivono l'utente come l'età e il sesso, a cui si aggiungo 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].

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 di tanh e sigmoide che hanno un output massimo uguale a 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

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

dove y è il valore reale del feedback utente (0 oppure 1), e $p(y)$ è la probabilità predetta dalla rete che y abbia valore 1.

Come ottimizzatore ho scelto Adam, nel paper in cui viene introdotto viene dimostrato empiricamente di essere generalmente migliore rispetto ad altri algoritmi di ottimizzazione stocastici e di risolvere in modo efficiente problemi di deep learning [6]. Adam ha diversi parametri configurabili, il più importante è il learning rate (chiamato α in Adam) che viene deciso tramite grid search, gli altri parametri $(\beta_1, \beta_2, \varepsilon)$ sono lasciati al valore di default della libreria Keras.¹ Gli altri iperparametri della rete neurale come il numero di epoche e la batch size sono ottimizzati tramite grid search e descritti nel Capitolo 6. In Figura 2 è rappresentata la struttura della rete. In questo caso la rete ha due layer nascosti ed ogni layer contiene 10 neuroni. Come si vede dall'input le feature di contesto sono più numerose rispetto a quelle di user, item e social context prese singolarmente. Ovviamente questo non è un constraint ma ci si aspetta che le feature di contesto fisico siano molto numerose.

3.4 Preprocessing dei dati

3.4.1 Contesto sociale

Il contesto sociale si riferisce all'insieme di persone con cui l'utente ha interazioni sociali durante la vita giornaliera, come lavorare con il colleghi o chattare con gli

¹<https://keras.io/api/optimizers/adam/>

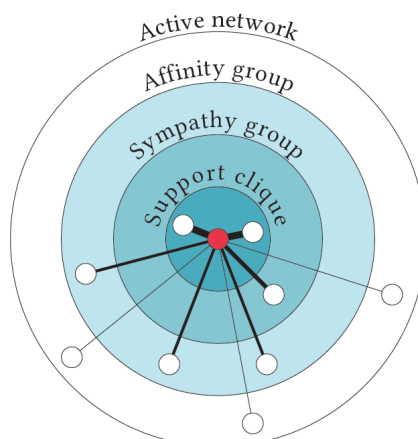


Figura 3: Ego network

amici. È stato provato in letteratura che esiste una forte correlazione tra le attività umane e i dati sociali. Questo implica che modellare una rete specifica per l'utente di relazioni sociali può contribuire a sottolineare le differenze tra i vari contesti in cui è coinvolto.

Ego Network

Una ego network è una rete sociale composta da un individuo chiamato ego, e dalle persone con cui l'ego ha un collegamento sociale, chiamati alter. I legami sociali in una ego network non hanno tutti la stessa importanza. Ogni individuo ha solo pochi collegamenti forti e molti collegamenti deboli, dovuti alla capacità umana di gestire un numero limitato di relazioni sociali. Una rappresentazione della ego network è mostrata in Figura 3: l'ego è il punto rosso al centro dei quattro cerchi concentrici chiamati layer in cui gli alter sono distribuiti in base alla forza del legame sociale con l'ego. Il cerchio più interno (support clique) è il layer più piccolo, e contiene solo pochi alter che rappresentano le relazioni sociali più forti con l'ego. Il secondo layer (sympathy group) contiene le persone che possono essere considerati gli amici più cari. Il terzo cerchio (affinity group) è composto da "casual friends" [tradurre] e membri della famiglia meno vicini, mentre l'ultimo layer include persone con cui l'individuo ha interazioni sociali occasionali.

Modellare il contesto sociale dell'utente

Per modellare il contesto sociale di un utente in ambiente mobile, si caratterizzano le interazioni sociali usando le seguenti sorgenti di dati: (i) chiamate telefoniche e log degli SMS, (ii) dati di prossimità, e (iii) attività svolte dall'utente sugli online social networks (OSN).

Il primo step per costruire l'ego network di un individuo è stimare la forza dei legami sociali con i suoi alter. Un buon indicatore della forza della relazione sociale tra due persone è data dal numero di interazioni che le due persone hanno avuto in passato. Basandosi su questa considerazione, per modellare la forza dei legami sociali dell'utente nel cyberspace[???che robba è], sono presi in considerazione diverse attività svolte dall'utente su OSN, inclusi commenti, reazioni (come "mi piace") e persone menzionate. Formalmente, la forza dei legami sociali virtuali tra l'ego e e uno dei suoi alter a , $\omega(e, a)$ è calcolata nel modo seguente:

$$\omega(e, a) = \sum_{v \in V} I_S(e, a)$$

dove V è l'insieme delle sorgenti di dati degli OSN nominate prima, e la funzione $I_S(e, a)$ calcola il numero di interazioni tra e ed a per una data sorgente di dati.

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.
- [6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.