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

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 per dispositivi mobili e pervasivi

3.1 Introduzione

In questo capitolo è descritto un sistema di raccomandazione context-aware per sistemi mobili e pervasivi deep learning che è in grado di generare raccomandazioni direttamente dal device dell'utente. Il RS è inserito all'interno di un architettura più complessa che permette sempre da device utente di raccogliere e processare le feature di contesto sociale e fisico. L'architettura ad alto livello è composta da tre componenti:

1. *Sensing manager.* Il primo componente interagisce con il sistema operativo per raccogliere continuamente dati raw di contesto (come GPS e accelerometro), e dati che rappresentano informazioni più astratte relative allo stato del dispositivo come lo stato del display e il livello di batteria.

2. *Context modeling.* I dati raw dei sensori devono essere processati ulteriormente per inferire una rappresentazione più astratta del contesto dell'utente. A questo scopo, il componente context modeling (CM) raccoglie periodicamente gli ultimi dati disponibili del SM. Queste osservazioni sono processate per estrarre feature numeriche e categoriche che caratterizzano il contesto dell'utente locale. Esempi di queste feature sono ad esempio le statistiche ottenute dai dati dei sensori fisici o la posizione dell'utente. Di queste feature eterogenee viene fatto l'encoding e vengono combinate in un singolo vettore di feature che rappresenta una fotografia del contesto corrente di un utente. L'insieme di feature che compongono il contesto fisico e sociale dell'utente sono descritte nella sezione 3.3.

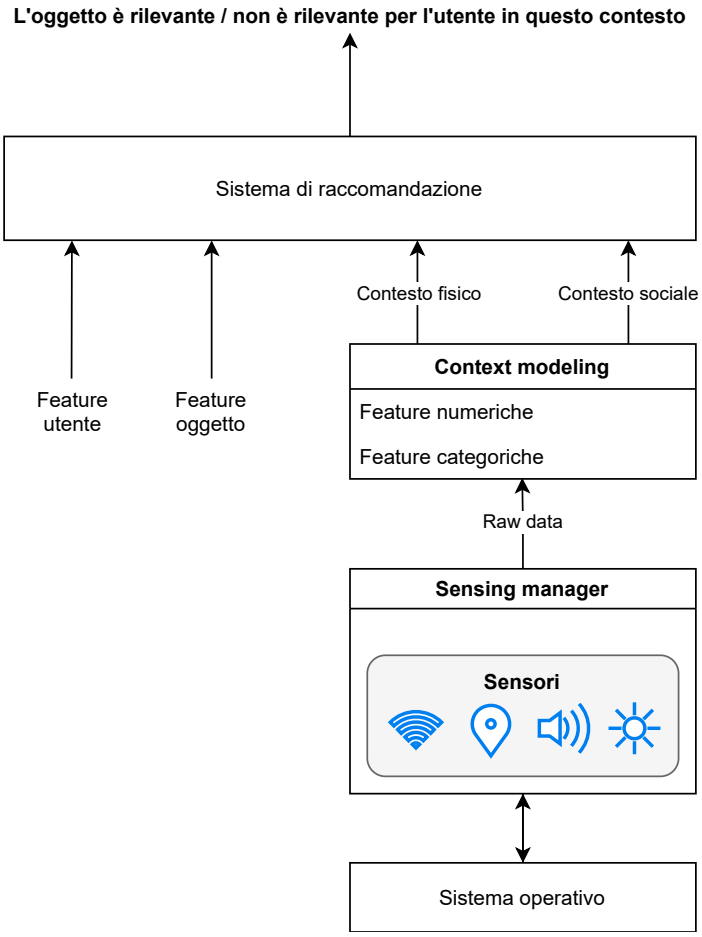

Figura 1: Architettura ad alto livello del sistema di raccomandazione

3. *Sistema di raccomandazione.* Le feature di contesto c sono concatenate alle feature degli utenti u e degli oggetti i in un unico vettore. Questo vettore è dato in input ha una rete neurale che restituisce valore 1, se per l'utente con feature u, l'oggetto con feature i è rilevante nel contesto c, 0 altrimenti. L'input, la struttura e l'output della rete sono descritti nella sezione sezione 3.2.

3.2 Sistema di raccomandazione

In questa sezione è descritto il sistema di raccomandazione vero e proprio. Nella prima parte è descritto l'input, e in che modo si differenzia dai sistemi di raccomandazione collaborative filtering e content-base. Nella seconda parte invece è

descritta nel dettaglio la struttura della rete neurale che genera le raccomandazioni context-aware.

3.2.1 Input

Solitamente l'input dei modelli collaborative filtering context-aware è composto da tuple `(user_ID, item_ID, rating, context)`, in cui `user_ID` è l'utente che ha valutato l'oggetto `item_ID` con una valutazione `rating` in una situazione descritta dal contesto `context`. Al posto che identificare l'oggetto con un valore numerico intero `item_ID`, si possono usare delle feature che caratterizzano l'oggetto, esattamente nello stesso modo in cui sono solitamente descritti gli oggetti nei sistemi di raccomandazione content-based. Ad esempio, se si sta sviluppando un RS per consigliare ristoranti agli utenti, si può sostituire il valore `item_ID` che identifica il ristorante con delle feature che lo caratterizzano nel dettaglio come il tipo di cibo servito, il prezzo medio, l'atmosfera, se ha sedute all'aperto, etc. Allo stesso modo si può sostituire il valore `user_ID` con delle feature che descrivono l'utente. Queste possono essere feature molto generiche come età o sesso, o feature specifiche per l'ambiente in cui il RS è implementato. Tornando all'esempio dei ristoranti, si potrebbe chiedere all'utente (es. attraverso un'applicazione mobile) 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 generate dal modulo di Context modeling. Un'istanza di rating per il modello proposto in questa tesi è quindi una tupla `(user_features, item_features, physical_context, social_context)`.

3.2.2 Struttura della rete neurale

Il vettore di feature appena descritto è dato in input ad una rete neurale che deve prevedere se l'utente caratterizzato da `user_feature` è interessato all'oggetto caratterizzato da `item_feature` nei contesti fisici e sociali `physical_context` e `social_context`. Si tratta quindi di un problema di classificazione binaria. Nei problemi di classificazione, l'obbiettivo è prevedere il valore di una variabile che può assumere diversi valori discreti. I problemi di classificazione in cui una variabile può assumere solo due valori possibili (come 0 o 1) sono chiamati problemi di classificazione binaria [1].

Layer e neuroni La rete rientra nella categoria feed-forward fully connected. Una rete feed-forward non contiene cicli nel suo grafo [2], 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 l layer nascosti. Il layer di input ha un numero di neuroni pari alle feature in ingresso (sommando user, item e context

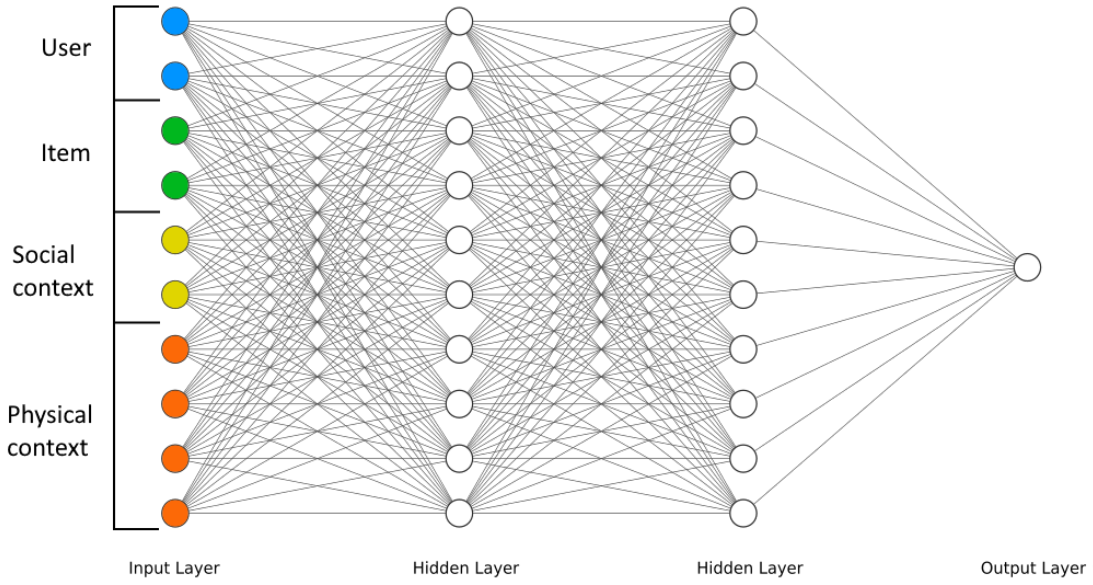

Figura 2: Schema di nome modello

feature), il layer di output ha sempre un neurone, mentre il numero di neuroni nei layer nascosti l, e il numero di layer nascosti è calcolato facendo il tuning della rete tramite grid search, scegliendo la combinazione che ottiene i risultati migliori. 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 contengono meno neuroni rispetto ai layer meno profondi.

Funzione di attivazione Una funzione di attivazione di un neurone definisce l'output di quel neurone in base all'insieme dei suoi input. 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 [2] 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 [3]. Come funzione di attivazione del layer di output ho scelto la funzione sigmoide definita come

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

che limita l'output della rete a valori tra 0 e 1, ed è quindi adatta per problemi di classificazione binaria [4].

Funzione di loss Una funzione di loss è una misura dell'errore tra il valore previsto dal modello e il valore effettivo. Come funzione di loss la scelta più comune per un classificatore binario è la funzione 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 di un utente su un oggetto (0 oppure 1), $p(y)$ è la probabilità predetta dalla rete che y abbia valore 1, e $1 - p(y_i)$ è la probabilità che y abbia valore 0.

Ottimizzatore 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 [5]. 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 tranne il layer di output che contiene un solo neurone.

3.3 Informazioni di contesto

In questa sezione sono descritte le informazioni di contesto fisico e sociale che vengono date in input al sistema di raccomandazione mobile. Tutte le feature sono raccolte e processate direttamente sul device dell'utente.

3.3.1 Contesto fisico

Il contesto fisico è composto da tutte quelle informazioni rilevanti che possono essere utilizzate per caratterizzare la situazione di un utente. Le feature del contesto fisico sono ricavate dai sensori fisici dello smartphone di un utente (es. attività utente dall'accelerometro) e dal sistema operativo del telefono (es. stato display e livelo batteria). A queste feature si vanno a integrare informazioni esterne come

[1] `https://keras.io/api/optimizers/adam/`

il meteo, la data e l'ora. Più nel dettaglio il contesto utente è caratterizzato dalle seguenti informazioni:

Posizione Informazioni relative alla posizione geografica che includono latitudine, longitudine, precisione della posizione e direzione del movimento. La posizione geografica può essere usata per capire il luogo in cui si trova l'utente (a casa, al lavoro, etc.) o per raccomandare punti di interesse nelle vicinanze.

Movimento utente Il movimento dell'utente include sia le attività svolte a piedi (correre e camminare), sia il movimento su un mezzo di trasporto (veicolo generico o bicicletta).

Audio Informazioni relative alla configurazione audio dello smartphone, incluse la modalità audio (suono, vibrazione, silenzioso), il volume delle notifiche, e lo stato dell'altoparlante (acceso o spento). Anche l'audio può migliorare il riconoscimento del contesto, per esempio durante una riunione la modalità audio potrebbe essere impostata su silenzioso e l'altoparlante spento.

Batteria Informazioni relative alla batteria del telefono che includono il livello di carica e se la batteria si sta ricaricando.

Display Stato dello schermo dello smartphone (acceso o spento), e orientamento dello schermo (verticale od orizzontale).

Dati dei sensori fisici che includono sensori ambientali (es. temperatura dell'ambiente e luce), sensori di movimento (es. accelerometro e giroscopio) e sensori di posizione (es. rotazione e prossimità).

Celle di rete Lista delle celle di rete mobile rilevate dal dispositivo. Per ogni cella si identifica il tipo di tecnologia (es. GSM o LTE), l'ID della cella, e la forza del segnale. La rete mobile può migliorare l'identificazione della posizione dell'utente.

Wi-Fi Lista di tutti gli access point Wi-Fi disponibili in prossimità, e se l'utente è connesso ad uno di essi.

Meteo Informazioni relative alle condizioni meteo che includono il tempo in atto (es. nuvoloso, piovoso, soleggiato), la temperatura, l'umidità e la velocità del vento.

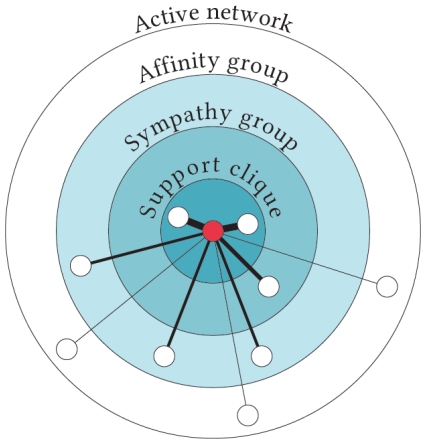

Figura 3: Ego network

Data e ora Dalla data si possono generare feature come il giorno della settimana, la stagione, comprendere se è il fine settimana o un periodo di vacanza, etc. Dall'orario invece si può capire il momento della giornata (mattina, pomeriggio, sera, notte).

3.3.2 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 i colleghi o chattare con gli amici. É 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 rappresentato 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 amici 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 delle relazioni 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 online, 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 ed uno dei sui alter a, $\omega_{osn}(e, a)$ è calcolata nel modo seguente:

$$\omega_{osn}(e, a) = \sum_{v_\in V} I_S(e, a) \tag{1}$$

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.

Per caratterizzare i link sociali fisici di un utente si calcola il numero di interazioni con altre persone basandosi su telefonate, SMS e contatti faccia a faccia inferiti usando tecnologie wireless disponibili sugli smartphone. In particolare sono considerate il Bluetooth (BT) e il Wi-Fi Direct (WFD) per scoprire persone che sono abbastanza vicine da aver un interazione con l'utente locale. Sono filtrati i dispositivi che non si trovano in prossimità dell'utente, e sono selezionati solo i dispositivi personali dell'utente, in modo tale da non considerare stampanti smart TV etc. In modo simile ai link sociali virtuali, si definisce la forza dei legami fisici sociali tra l'ego e e un alter a , $\omega_{phy}(e, a)$ come il numero delle loro interazioni tramite telefonate, SMS, e prossimità fisica come segue:

$$\omega_{phy}(e, a) = \sum_{p\in P} I_p(e, a) \tag{2}$$

dove P è l'insieme delle sorgenti fisiche considerate, e $I_p(e, a)$ rappresenta il numero di interazioni tra due utenti per una data sorgente di dati. Infine, la forza complessiva del collegamento sociale tra e ed a è data dalla combinazione lineare delle interazioni online e fisiche descritte prima:

$$\omega_s(e, a) = \lambda \cdot \omega_{osn}(e, a) + (1 - \lambda) \cdot \omega_{phy}(e, a) \tag{3}$$

con il parametro λ che regola l'importanza delle interazioni sociali e fisiche. Per ogni alter, solo l'ultimo peso calcolato è mantenuto in memoria, e viene aggiornato quando nuove interazioni sociali sono identificate. I link sociali tra l'utente locale e le altre persone sono raggruppate in base al peso calcolato nell'Equazione 3. L'output finale è un array di valori in cui ogni elemento rappresenta la percentuale di utenti attivi in ogni cerchio della ego network di un utente.

Capitolo 4

Capitolo 4

Capitolo 5

Capitolo 5

Capitolo 6

Conclusioni

6.1 Conclusioni

Conclusioni...

6.2 Sviluppi futuri

Sviluppi futuri...

Bibliografia

[1] Aurélien Géron. Hands-on machine learning with scikit-learn and tensorflow: Concepts. *Tools, and Techniques to build intelligent systems*, 2017.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[3] Xavier Glorot, Antoine Bordes, and Y. Bengio. Deep sparse rectifier neural networks. volume 15, 01 2010.

[4] Jason Brownlee. How to choose an activation function for deep learning. `https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/`, 2021.

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