Deep Learning

Università Roma Tre
Dipartimento di Ingegneria

Anno Accademico 2022 - 2023

Recommender Systems

Sommario

- MovieLens dataset
- AutoRec
- Implicit feedback (richiami)ù
- Neural Collaborative Filtering (NCF)
- Caser e Sequence-aware Recsys
- Factorization Machines e Deep FM

Motivazioni

- I **sistemi di raccomandazione** sono strumenti chiave in molti scenari: e-commerce, siti per la fruizione di musica e video (es. Spotify, Netflix), app stores, pubblicità, etc.
  - Alcune conferenze internazionali specifiche nel settore (es. RecSys) attraggono i maggiori players che fanno a gara per aggiudicarsi i migliori ricercatori.
- Supponiamo nel resto dei lucidi che alcuni argomenti sia già noti dai precedenti corsi:
  - Collaborative filtering
  - Explicit e Implicit feedback
  - Recommendation tasks (es. rating vs top-n vs sequence aware recommendation

MovieLens dataset

- Il dataset più popolare nel mondo accademico **RecSys**. Ne esistono diverse versioni in base alla quantità di dati contenuti.
  - In **MovieLens 100K** sono contenuti 100.000 ratings espressi in una scala da 1 a 5, da 943 utenti su 1682 film. Ogni utente ha espresso rating su almeno 20 film. Il formato del dataset è **csv**.
  - http://files.grouplens.org/datasets/movielens/ml-100k.zip

- Tecniche quali **Matrix Factorization** sono state in in grado di individuare patterns chiave per ottenere migliori risultati rispetto ad approcci più tradizionali. Ma si limitano a catturare patterns e correlazioni di tipo lineare.

MovieLens dataset

- MovieLens **100K** vs **1M**

	ML-100K	ML-1M
Number of users	943	6,040
Number of movies	1,682	3,952
Number of ratings	100,000	1,000,209
Number of all genres	19	18
Average number of genres	1.7	1.6
Rating scales	1–5	1–5

AutoRec

- In **AutoRec** si impiega un paradigma di Collaborative Filtering basato su autoencoders. Invece di rappresentare la matrice user-ratings in un spazio latente, o di impiegare le stesse istanze di input anche per l'output come nel caso degli autoencoders visti in precedenza, si segue un approccio alternativo:
  - in **input** si hanno un le interazioni utente-item osservate
  - in **output** ci si aspetta l'intera matrice delle interazioni utente-item

- Esistono architetture AutoRec *user-based* e *item-based*. Qui vedremo le seconde ma è facile immaginare le prime per analogia.

(Item-based) AutoRec

- Indichiamo con R_{*i} la i-ma colonna della matrice dei ratings. I valori di rating sconosciuti saranno 0.
- La rete AutoRec la possiamo definire formalmente:

$$h(R_{*i}) = f(W \cdot g(V \cdot R_{*i} + \mu) + b)$$

- dove f e g sono funzioni di attivazione, W e V matrici di pesi, μ e b sono biases, $h(R_{*i})$ la ricostruzione della i-ma colonna.
- La funzione da ottimizzare è la seguente:

$$\text{argmin}_{W,V,\mu,b} \sum_{i=1}^{M} ||R_{*i} - h(R_{*i})||^2 + \lambda(||W||_F^2 + ||V||_F^2))$$

- dove il primo modulo considera solo i *ratings* noti.

Richiami: Implicit Feedback

- I *rating espliciti* (es. valutazioni da 1 a 5) sono generalmente scarsi nei servizi di raccomandazione. Inoltre valori mancanti di rating possono essere erroneamente interpretati, es.: forme di feedback negativi invece di rating che devono essere ancora specificati.
- Si tendono a sfruttare altre fonti che possono essere interpretate come forme di *implicit feedback*.
  - Es. clicks, acquisti, visite, wish lists.

Neural Collaborative Filtering for Personalized Ranking

- Il **Neural Collaborative Filtering (NCF)** *framework con implicit feedback* sfrutta la capacità di *non-linearità* delle reti neurali.
- È composto da 2 reti, una basata su Generalized Matrix Factorization, l'altra consiste in una MLP.
- L'output delle 2 reti è concatenato per generale la predizione finale. Se in AutoRec puntavamo a predire i rating, NCF produce una lista di raccomandazioni con score associato ad ogni item della lista.

Neural Collaborative Filtering for Personalized Ranking

- La **GMF** è un approccio di MF implementato con reti neurali. L'input è il prodotto element-wise (Hadamard product) tra le rappresentazioni latenti degli utenti e degli item:

$$\mathbf{x} = \mathbf{p}_u \odot \mathbf{q}_i$$
$$\hat{y}_{ui} = \alpha(\mathbf{h}^\top \mathbf{x}),$$

- Dove $\boldsymbol{p_u}$ e $\boldsymbol{q_i}$ sono rispettivamente la u-ma riga di P e q-ma riga di Q, dove $P \in \mathbb{R}^{m \times k}$ e $Q \in \mathbb{R}^{n \times k}$. L'output è la previsione di score dell'utente u per l'item i.

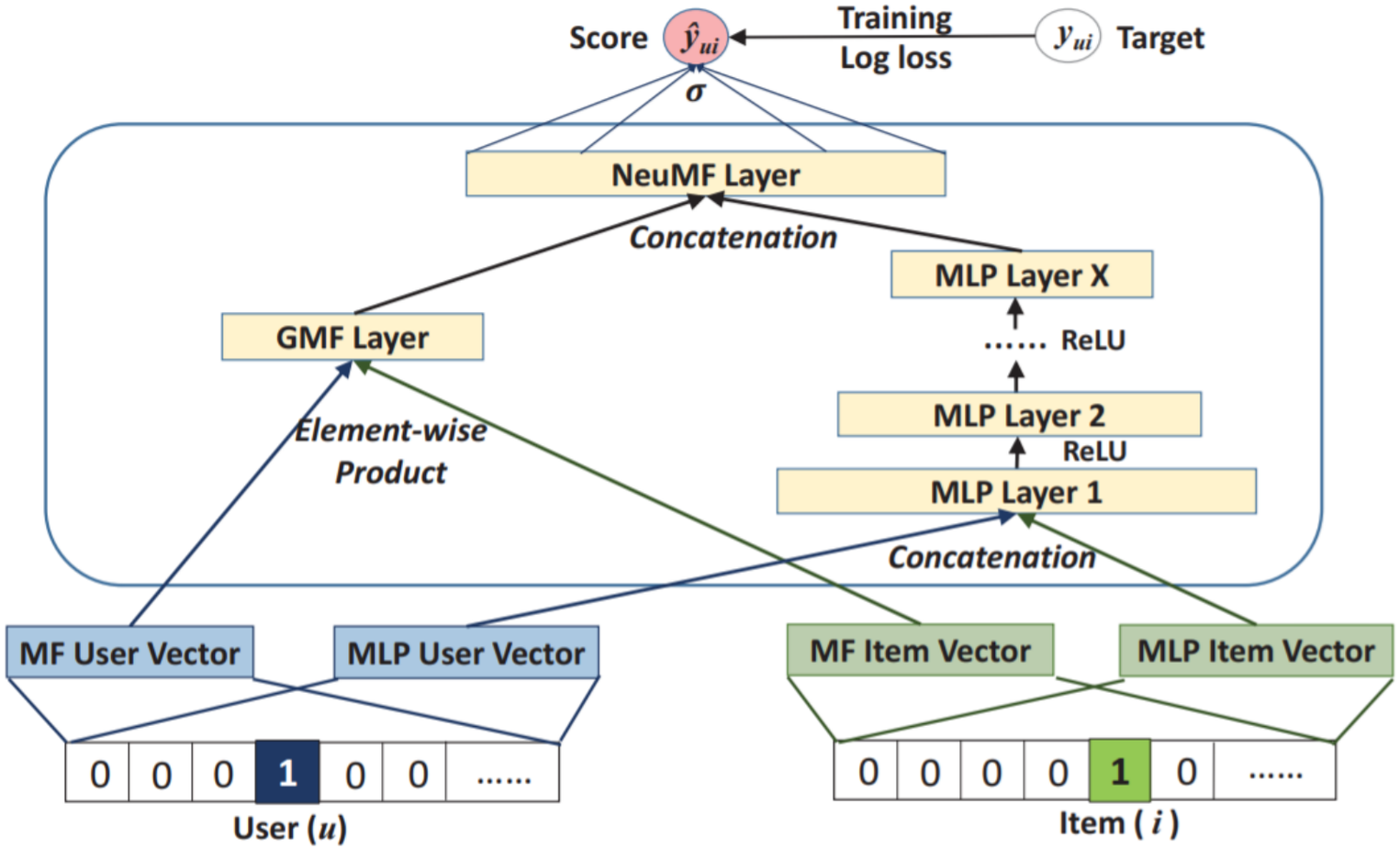
- La **MLP** prende in input le rappresentazioni degli utenti e item, ignorando lo spazio latente della GMF. Lo scopo è individuare correlazioni aggiuntive con operazioni non lineari.

$$z^{(1)} = \phi_1(\mathbf{U}_u, \mathbf{V}_i) = [\mathbf{U}_u, \mathbf{V}_i]$$
$$\phi^{(2)}(z^{(1)}) = \alpha^1(\mathbf{W}^{(2)} z^{(1)} + b^{(2)})$$
$$\dots$$
$$\phi^{(L)}(z^{(L-1)}) = \alpha^L(\mathbf{W}^{(L)} z^{(L-1)} + b^{(L)}))$$
$$\hat{y}_{ui} = \alpha(\mathbf{h}^\top \phi^L(z^{(L-1)}))$$

Neural Collaborative Filtering for Personalized Ranking

- L'output del penultimo layer di entrambe le reti è concatenato è dato in input al NeuMF layer:

$$\hat{y}_{ui} = \sigma(\mathbf{h}^\top[\mathbf{x}, \phi^L(z^{(L-1)})])$$



Sequence-aware recommender systems

- Spesso gli utenti operano una sequenza di azioni nei servizi online il cui ordine temporale può essere significativo nel processo di raccomandazione.
- Il modello **Caser** (Convolutional sequence embedding recommendation model) sfrutta le CNN, in particolare *horizontal* e *vertical* convolutional networks, per identificare rispettivamente pattern sequenziali *union-level* e *point-level* di tipo short-term.
- I pattern *point-level* identificano l'influenza di un item all'interno di una sequenza verso un certo item target. L'union-level analizza l'influenza di varie azioni fatte sul valore target (es. l'acquisto di latte e burro può implicare l'acquisto di farina).
- I bisogni di lungo termine sono rappresentati nei layer FC finali.

Sequence-aware recommender systems

- Supponiamo che ogni utente u sia associato ad una sequenza di items $S^u = (S_1^u, \cdots, S_{|S_u|}^u)$. Se prendiamo i passati *L* items, possiamo costruire una matrice che rappresenta le interazioni passati con tali items rispetto al time step ***t***.

$$\mathbf{E}^{(u,t)} = [\mathbf{q}_{S_{t-L}^u}, \ldots, \mathbf{q}_{S_{t-2}^u}, \mathbf{q}_{S_{t-1}^u}]^\top$$

- Dove $Q \in \mathbb{R}^{n \times k}$ rappresenta sempre lo spazio di embedding e $\mathbf{q_i}$ indica la *i*-ma riga. $E^{(u,t)} \in \mathbb{R}^{L \times k}$ è usata per ottenere i bisogni transienti dell'utente *u* per il time step *t*, ed è l'input per i successivi convolutional layers:
  - L'horizontal layer ha *d* filtri orizzontali $F^j \in \mathbb{R}^{h \times k}, i \leq j \leq d, h = \{1, \cdots, L\}$.
  - Il vertical layer ha d' filtri verticali $G^j \in \mathbb{R}^{L \times 1}, i \leq j \leq d'$
- Dopo una serie di operatori convoluzionali e di pooling otteniamo gli output $o \in \mathbb{R}^d$ e $o' \in \mathbb{R}^{d'}$:

$$\mathbf{o} = \mathrm{HConv}(\mathbf{E}^{(u,t)}, \mathbf{F})$$
$$\mathbf{o}' = \mathrm{VConv}(\mathbf{E}^{(u,t)}, \mathbf{G}),$$

Sequence-aware recommender systems

- Gli output sono concatenati e dati in input ad una MLP per ricavarne rappresentazioni ad alto livello:

$$\mathbf{z} = \phi(\mathbf{W}[\mathbf{o}, \mathbf{o}']^\top + \mathbf{b})$$

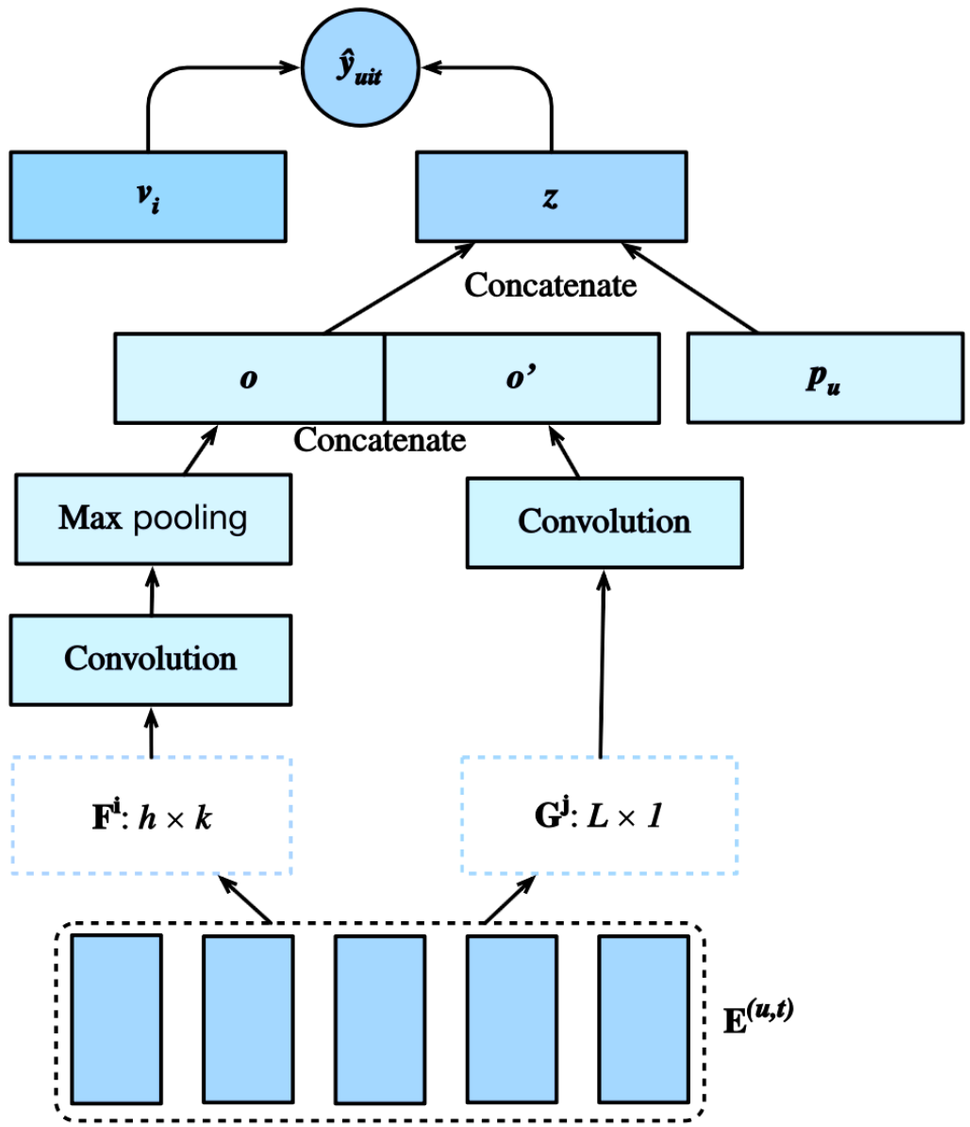
- L'output $\mathbf{z} \in \mathbb{R}^k$ indica i bisogni a breve termine dell'utente.
- I bisogni a lungo termine dell'utente sono ricavati:

$$\hat{y}_{uit} = \mathbf{v}_i \cdot [\mathbf{z}, \mathbf{p}_u]^\top + \mathbf{b}'_i$$

- dove $\mathbf{V} \in \mathbb{R}^{n \times 2k}$ è una ulteriore embedding matrix, e $\mathbf{P} \in \mathbb{R}^{m \times k}$ è la *user embedding matrix* per i bisogni a lungo termine. $\mathbf{p_u} \in \mathbb{R}^k$ e $\mathbf{v_i} \in \mathbb{R}^{2k}$ sono la *u*-ma riga e *i*-ma riga rispettivamente di $\mathbf{P}$ e $\mathbf{V}$.

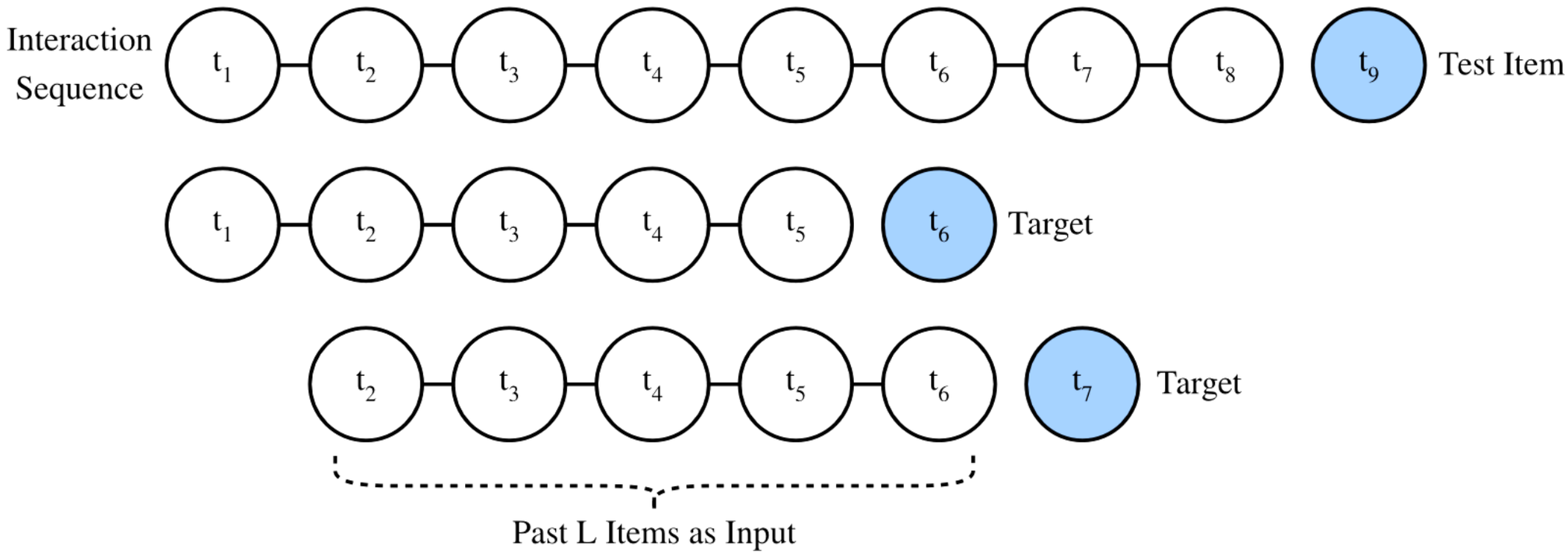
Sequence-aware recommender systems

- L'architettura generale di Caser è così rappresentata:



Addestramento e architetture sequenziali

- In modo simile all'addestramento RNN, in presenza di dati con timestamp che rappresentano azioni tra utenti e items (es. l'utente ha lasciato un rating su un film), possiamo costruire un dataset di addestramento creando sequenze di dimensione predefinita, ad esempio:



Factorization Machines

- Le FM sono algoritmi supervisionati che possono essere impiegati in contesti di classificazione, regressione e ranking. Si ispirano a modelli di regressione lineare, modelli di MF e Support vector machines con kernel polinomiali.
- Mostrano vantaggi in caso di dataset sparsi riducendo notevolmente il tempo di addestramento. Inoltre individuano più facilmente correlazioni significative tra i dati.
- Se indichiamo con $x \in \mathbb{R}^d$ il vettore delle feature per una certa istanza, e con y la label numerica associata (es. 4.5 oppure click/no-click), formalizziamo il modello in questo m

$$\hat{y}(x) = \mathbf{w}_0 + \sum_{i=1}^{d} \mathbf{w}_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j$$

- dove $w_0 \in \mathbb{R}$ è il bias globale, $\mathbf{w} \in \mathbb{R}^d$ sono i pesi per la i-ma variabile, $\mathbf{V} \in \mathbb{R}^{d \times k}$ sono i feature embeddings, e $\mathbf{v_i}$ è la i-ma riga di $\mathbf{V}$, $\langle v_i, v_j \rangle$ è il prodotto vettoriale tra i 2 vettori è modella l'interazione tra la *i*-ma e *j*-ma feature.

Factorization Machines

- Nell'espressione precedente si può notare una prima componente che rappresenta il modello di regressione lineare, mentre il secondo estende un modello di MF:

$$\hat{y}(x) = \mathbf{w}_0 + \sum_{i=1}^{d} \mathbf{w}_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j$$

- Se la feature ***i*** rappresenta un item, e la feature ***j*** un utente, il terzo termine è il prodotto scalare tra i due embedding, uno dell'utente ed uno dell'item.

Deep Factorization Machines

- In alcuni scenari l'approccio lineare delle FM non è sufficiente per rappresentare correlazioni e patterns complessi. Ma è possibile integrare approcci "deep" nel FM per rappresentare interazioni tra features nei dati.
- Nel **DeepFM** la componente FM e deep sono combinate in modo *parallelo*. La FM è simile all'architettura originale. La deep è implementata con una MLP. L'input/embeddings delle 2 componenti è il medesimo, l'output è *sommato* per creare la predizione finale.
- La DeepFM si ispira allearchitetture RecSys chiamate **wide & deep**. In tali architetture la predizione combina due pipeline:
  - la *memorizzazione* mira a rappresentare co-occorrenze frequenti tra items o features nei dati storici. Si implementa con un modello lineare (es. logistic regression)
  - la *generalizzazione* punta a implementare la "transitività" delle correlazioni, cioè esplorare combinazioni significative tra features che non sono state mai incontrate nel passato. La pipeline è generalmente basata su una MLP.

Deep Factorization Machines

- Supponiamo che l'output della FM sia $\hat{y}^{(FM)}$. Indichiamo con $\mathbf{e_i} \in \mathbb{R}^k$ il vettore delle feature latenti del campo *i*-mo.
- L'input della componente deep è la concatenazione degli embeddings di tutti i campi associati alle *f* feature categoriche date in input:

$$\mathbf{z}^{(0)} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_f]$$

- La rete neurale è cosi definita:

$$\mathbf{z}^{(l)} = \alpha(\mathbf{W}^{(l)}\mathbf{z}^{(l-1)} + \mathbf{b}^{(l)})$$

- L'output $\hat{y}^{(DNN)}$ è combinato con il precedente per generare l'output finale:

$$\hat{y} = \sigma(haty^{(DNN)} + \hat{y}^{(DNN)})$$

Deep Factorization Machines

- L'architettura DeepFM è la seguente: