

Recommendation

Programmazione di Applicazioni Data Intensive

Laurea in Ingegneria e Scienze Informatiche

DISI – Università di Bologna, Cesena

Proff. Gianluca Moro, Roberto Pasolini

nome.cognome@unibo.it



Recommendation: Una Storia Vera (i)

- Minneapolis, Marzo 2012, Supermercati TARGET
 - Un padre un po' arrabbiato si presenta in una filiale della catena chiedendo di parlare con un manager
 - *"mia figlia riceve da settimane vostri coupon sconto su prodotti per la maternità come vestiti, culle e pannolini per neonati ... ma sta facendo ancora la scuola superiore, la state incoraggiando a rimanere incinta?"*
 - Il manager controlla la posta inviata e rileva che il materiale indirizzato alla figlia dell'uomo conteneva coupon sconto e pubblicità per abbigliamento e articoli premaman, ma anche foto di neonati sorridenti ...
 - Il manager: *"ci scusi deve esserci stato un errore"* e alcuni giorni dopo richiama il padre per scusarsi nuovamente
 - Al telefono, però, il padre era piuttosto imbarazzato. *"Ho parlato con mia figlia ... ci sono state alcune attività in casa mia di cui non ero consapevole. Lei partorirà in Agosto. Le devo io delle scuse"*



fonte: Forbes - Charles Duhigg of New York Times



Recommendation: Una Storia Vera (ii)

- Come il supermercato ha previsto che la ragazza era incinta e quali prodotti sarebbero stati più appropriati per lei ?
 - il supermercato associa ad ogni scontrino un codice cliente che dipende dalla carta di credito, dall'email o dall'indirizzo
 - perciò ha lo **storico degli acquisti** di ogni cliente
 - ed anche **dati demografici** acquisiti direttamente o acquistati
- Le spese per neonati e bambini sono un grande business
 - il supermercato avviò anni prima un progetto mirato ad “agganciare” i genitori imminenti, prima che diventino clienti della concorrenza
- Cos'è stato scoperto dall'analisi dei dati ?
 - le donne incinta (*stato desumibile anche a posteriori dall'età del figlio*) acquistano di più lozioni inodore all'inizio del 2°trimestre di gravidanza
 - nelle prime 20 settimane anche integratori di calcio, magnesio e zinco
 - l'aumento del consumo di battufoli di cotone, disinfettanti per le mani e salviette indicano che sono prossime al parto

Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

3

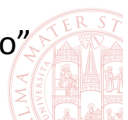


Recommendation: Una Storia Vera (iii)

- **C'è di più ...**
 - sono stati identificati circa 25 prodotti che insieme predicono addirittura una stima della data di nascita in un piccolo intervallo
 - → propongono coupons diversificati in base alla fase della gravidanza
 - uno score predice lo stato di gravidanza in base agli acquisti:
 - e.g. donna di 23 anni, a **Marzo** raddoppia il consumo di lozioni inodore, magnesio, zinco, acquista un tappeto blu brillante -> **parto in Agosto 87%**
- il supermercato ha rilevato anche un comportamento inatteso
 - un alto numero di futuri genitori che ricevevano solo materiale pubblicitario per la maternità non diventavano clienti
 - distribuendo invece articoli di maternità in mezzo ad altri, procura maggiori vendite
 - Perché ? la conclusione fu che a nessuno piace essere troppo “spiato”
- Aumentato il fatturato del 50%

Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

4



Sistemi di Raccomandazione

- I Sistemi di raccomandazione, i.e. Recommender Systems (RS), mirano ad individuare relazioni tra utenti e prodotti
 - forniscono informazioni, i.e. raccomandazioni, personalizzate per l'utente, assistenza vendite (orientamento, consulenza, persuasione...)
- Macro-famiglie di approcci
 - **Sistemi Collaborativi**: “Dimmi ciò che è popolare tra gli utenti con interessi simili ai miei”
 - **Sistemi Content-based**: “Mostrami oggetti simili per contenuto a ciò che ho apprezzato in passato” e.g. individua libri simili per contenuto a quelli che ho letto
 - **Sistemi Knowledge-based**: “Dimmi quello che si adatta a me in base alle mie esigenze” e.g. l'utente esprime le caratteristiche del prodotto di interesse e il recommender cerca i prodotti che più le soddisfano
 - **Sistemi Ibridi**: combinazioni delle tecniche precedenti

Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

5



La Long Tail

- Nella vendita on-line il num. di prodotti esistenti è molto superiore al numero di prodotti che l'utente può analizzare
 - occorre individuare quelli con maggiore propensione di acquisto per l'utente
- La raccomandazione nel mondo fisico è in generale più semplice
 - I prodotti sono fisicamente molti di meno (solo i più popolari) e l'utente può visionarli tutti o quasi
- fenomeno “long tail”: ossia pochi prodotti sono acquistati/visionati molto e molti prodotti sono acquistati/visionati poco
 - fenomeno come tanti altri che segue una distribuzione power law



Su MovieLens il 74% di tutte le valutazioni positive sono fatte sul 20% dei prodotti

Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

6



Perchè usare Sistemi di Raccomandazione?

- **Valore per il cliente**
 - Trovare oggetti rilevanti rispetto ai propri interessi
 - Migliorare l'insieme delle scelte
 - Aiutare ad esplorare lo spazio di opzioni
 - Intrattenimento
- **Valore per il provider**
 - Servizio personalizzato aggiuntivo per i clienti
 - Aumentare la fiducia e la fidelizzazione dei clienti
 - Aumentare le vendite, mediante aumento di click through rates (percentuale di click per misurare il gradimento di oggetti proposti)
 - Opportunità per la promozione, la persuasione
 - Ottenere maggiore conoscenza sui clienti in base al gradimento delle proposte fatte dal recommender

Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

7



Nel Mondo Reale

- **Amazon.com** e **Netflix** (noleggio DVD e film in streaming) generano tra il 30% e il 70% delle vendite attraverso le liste di raccomandazione
- Ma non solo....
 - Raccomandazione di gruppi, posti di lavoro o persone su **LinkedIn**
 - Raccomandazione di amici e personalizzazione delle pubblicità su **Facebook**
 - Raccomandazione di canzoni su **last.fm** o **spotify**
 - Raccomandazione di notizie su **Forbes.com** (più 37% Click-through rate grazie al recommender)
- **Studi scientifici**
 - Diversi studi mostrano l'aumento delle vendite e cambiamenti nel comportamento delle vendite grazie ai recommender

Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

8



Netflix Grand Prize Contest

- Un esempio dell'importanza nel mondo reale dei sistemi di raccomandazione è il *Netflix Grand Prize Contest*:
- Training:
 - 17,770 film / 480,189 utenti
 - >100 milioni di valutazioni (in scala 1-5, con timestamp)
- Test:
 - prevedere il voto di ogni tripla con voto ignoto (utente, film, voto ?)
 - 4.2 milioni di valutazioni
- Goal:
 - Migliorare di almeno il 10% la capacità predittiva del sistema di raccomandazione del sito Netflix (radice dell'errore quadratico medio, RMSE - vediamo dopo)
- Premio: **1 milione di \$** - *il vincitore ha usato SVD che vediamo dopo*

Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

9



La Utility Matrix

- In un Recommender System agiscono due attori principali
 - Utenti
 - Prodotti, news, posts, ristoranti, viaggi, offerte di lavoro ...
- I dati sono organizzati in una matrice, detta **utility matrix** (o matrice di rating) in cui:
 - ogni riga corrisponde ad un utente
 - ogni colonna corrisponde ad un prodotto
 - ogni cella corrisponde alla valutazione dell'utente per il prodotto, oppure può essere vuota se l'utente non si è espresso sul prodotto
- La matrice può essere binaria (0/1, se l'utente ha comprato o meno il prodotto) oppure discreta (es. Valori da 1 a 5)
- La matrice è generalmente molto sparsa
 - ogni utente dà voti ad una piccola parte dei prodotti

Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

10



Un esempio ...

- Utility Matrix che rappresenta valutazioni di film su una scala 1-5 (con 5 il punteggio più alto)
- Le celle vuote rappresentano la situazione in cui l'utente non ha valutato il film
- I nomi di film sono HP1, HP2 e HP3 per Harry Potter I, II, e III, TW per Twilight, e SW1, SW2 e SW3 per Star Wars episodi 1, 2, e 3.

	HP1	HP2	HP3	TW	SW1	SW2	SW3
Jake	4			5	4		
Mike	5	5	4				
Carl				2	4	5	
Tom		3		1	4	4	5

Carl apprezzerà o meno Star Wars 3?



Tipi di giudizi: Giudizi **Espliciti**

- rappresentano i feedback con maggiore probabilità di precisione
- Sono i più comunemente utilizzati (es: da 1 a 5, da 1 a 7, etc.)
- Aspetti tutt'ora di studio
 - Granularità ottimale di scala
 - es: recommendation di film la scala 0-10 si è dimostrata migliore
 - Valutazioni multidimensionali (più voti per ogni film, come le valutazioni per gli attori, le colonne sonore, ecc.)
- Problemi principali
 - Gli utenti non sempre sono disposti a votare molti prodotti
 - numero di voti a disposizione potrebbe essere troppo piccolo → Utility Matrix sparse di scarsa qualità
 - Problema: Come incentivare gli utenti a valutare più articoli ?



Tipi di giudizi: Giudizi **Impliciti**

- Tipicamente raccolti dal negozio web o applicazioni in cui il sistema di raccomandazione è incorporato
 - Es: l'acquisto di un prodotto è considerata una valutazione positiva (non sempre vero)
 - più sofisticati: Click, pagine viste, tempo speso sulle pagine, download di demo
- Possono essere raccolti costantemente e non richiedono ulteriori sforzi dal lato dell'utente
- **Problema:** Non si può essere certi che il comportamento degli utenti sia interpretato correttamente
 - Es: un utente potrebbe non gradire tutti i libri comprati
 - potrebbe anche aver comprato libri per qualcun altro
- Utilizzo dei giudizi impliciti insieme a quelli espliciti
 - con la possibilità di domandare all'utente una conferma sulla correttezza dell'interpretazione



Misurare la Bontà della Raccomandazione

- Esistono diverse metriche di misurazione dell'error rate
 - Mean Absolute Error (**MAE**): calcola la deviazione tra giudizi previsti p_i e quelli reali r_i

$$MAE = \frac{1}{n} \sum_{i=1}^n |p_i - r_i|$$

maggiore è la deviazione minore è l'accuratezza della raccomandazione

- Root Mean Square Error (**RMSE**): simile al MAE, ma pone maggiormente l'accento sulla maggiore deviazione

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (p_i - r_i)^2}$$



Sistemi Collaborativi

- Insieme di approcci chiamati di **Collaborative Filtering** (CF)
- Sono generalmente i più utilizzati
 - Utilizzati da grandi siti di e-commerce
 - applicabili di fatto in ogni settore commerciale (libri, film, news ...)
 - in letteratura esistono vari algoritmi e numerose varianti
- **Razionale:** fare recommendation utilizzando la *saggezza della massa*
- **Assunzione di base:** sono disponibili i voti degli utenti per gli articoli del catalogo (implicitamente o esplicitamente)
- **Ipotesi:** I clienti che hanno avuto interessi simili in passato, avranno interessi simili in futuro



Collaborative Filtering: Approcci Classici

- Input
 - La Utility Matrix del rating Utenti-Prodotti
- Output
 - Una previsione di quanto l'utente gradirà o meno un determinato prodotto
 - Un elenco ordinato dei top-N elementi consigliati per l'utente
- Approcci:
 - **Memory-based:** utilizzano **direttamente i dati** come gusti, voti, click, etc per rilevare correlazioni tra utenti (o elementi) e raccomandare ad un utente u un oggetto che non ha valutato/acquistato
 - **Model-based:** l'obiettivo è il medesimo ma utilizzano *algoritmi di apprendimento automatico* per la creazione di **modelli di learning** per fare **recommendation** ad ogni utente



User-based Nearest-Neighbor CF (I)

- Considerato un *utente*, vogliamo predire il gradimento di un prodotto *p* che non ha votato/comprato
 - Trovare il set di utenti più simili a *Carl* in base ai voti sugli stessi prodotti, ma che hanno votato il prodotto *p* non votato da *Carl*
 - Utilizzare, ad esempio, la media dei voti assegnati da questo set di utenti al prodotto *p* per **predire** il voto di Carl
 - Si ripete questa operazione per tutti i prodotti non ancora votati da *Carl* e si raccomanda quello con voto maggiore
- **Ipotesi:** *Se gli utenti in passato hanno votato/acquistato in maniera simile, lo faranno anche in futuro*
 - Le preferenze degli utenti sono considerate invarianti nel tempo
 - ciò che era piaciuto in passato ad un altro utente, può piacere ora all'utente che deve ricevere la recommendation



User-based Nearest-Neighbor CF (II)

- Come si misura la similarità tra utenti ?
- Quanti utenti simili dovremmo prendere in considerazione ?
- Come generiamo una previsione partendo dal rating degli utenti considerati ?

	Item1	Item2	Item3	Item4	Item5
Carl	5	3	4	4	?
Mike	3	1	2	3	3
Jake	4	3	4	3	5
Tom	3	3	1	5	4
Phill	1	5	5	2	1



Misurare la Similarità tra Utenti con Pearson - Correlazione Lineare

- Dati (assunzione: le variabili hanno distribuzione gaussiana)
 - a, b : sono due utenti
 - $r_{a,p}$ e $r_{b,p}$: rispettivamente rating dell'utente a e dell'utente b per il prodotto p
 - P : set di prodotti per cui sia a che b hanno espresso un giudizio
 - \bar{r}_a e \bar{r}_b : valore medio rispettivamente dei giudizi dell'utente a e b per i prodotti in P

- Correlazione di Pearson** (la più utilizzata):

$$sim_pearson(a,b) = \frac{\sum_{p \in P} (r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P} (r_{a,p} - \bar{r}_a)^2} \sqrt{\sum_{p \in P} (r_{b,p} - \bar{r}_b)^2}}$$

covarianza delle
variabili r_a ed r_b

prodotto delle 2
deviazioni standard

- Il risultato è compreso tra -1 ed 1:
 - se è maggiore di zero, la correlazione tra le due variabili è positiva
 - se è zero, non c'è correlazione
 - se è minore di zero, c'è correlazione inversa tra le variabili
- Spearman** per correlazioni NON lineari e variabili NON gaussiane

Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

19



Misurare la Similarità tra Utenti: Coseno

- Dati:
 - a, b : sono due utenti
 - $r_{a,p}$ e $r_{b,p}$: rispettivamente rating dell'utente a e dell'utente b per il prodotto p
 - P : set di prodotti per cui sia a che b hanno espresso un giudizio
 - r_a : vettore dei rating dell'utente a dei prodotti P

- Similarità coseno:**

$$sim_cosine(a,b) = \frac{\sum_{p \in P} (r_{a,p} \cdot r_{b,p})}{\sqrt{\sum_{p \in P} (r_{a,p})^2} \sqrt{\sum_{p \in P} (r_{b,p})^2}}$$

prodotto scalare
vettori r_a ed r_b

prodotto delle
norme dei 2 vettori

- Risultati della similarità coseno:
 - valori compresi tra 0, max dissimilarità, e 1 max similarità
- Altre misure di similarità, es: Distanza di Jaccard



Misurare la Similarità: Esempio

- Data la matrice dell'esempio precedente
- Misuriamo ed ordiniamo gli utenti simili a **Carl** secondo l'indice di correlazione di Pearson
- il più simile a Carl è **Mike**, segue Tom

	Item1	Item2	Item3	Item4	Item5
Carl	5	3	4	4	?
Mike	3	1	2	3	3
Jake	4	3	4	3	5
Tom	3	3	1	5	4
Phill	1	5	5	2	1



sim = 0,85
 sim = 0,00
 sim = 0,70
 sim = -0,79



Generare una Raccomandazione

- sia N l'insieme degli utenti più simili all'utente a
- $(r_{b,p} - \bar{r}_b)$: \forall utente $b \in N$, calcoliamo la **differenza** tra il giudizio su p di b e il giudizio medio di b sui prodotti valutati dall'utente a e b
- sia $sim(a,b)$ la similarità tra gli utenti a e b che serve a pesare il giudizio differenza di b usando come peso la similarità con a
- \bar{r}_a è la media dei giudizi espressi da a
- Per predire il prodotto p da raccomandare all'utente a si utilizza la seguente formula:

$$pred(a,p) = \bar{r}_a + \frac{\sum_{b \in N} sim(a,b) \cdot (r_{b,p} - \bar{r}_b)}{\sum_{b \in N} sim(a,b)}$$



Migliorare le Metriche di Predizione

- Non tutte le valutazioni degli utenti simili sono ugualmente utili
 - Essere d'accordo su oggetti comunemente apprezzati non è così informativo come un accordo sugli elementi controversi
 - Possibile soluzione: dare maggior peso agli elementi sui quali esiste maggiore varianza sui relativi voti
- Importanza del numero di elementi **co-rated**, i.e. apprezzati da entrambi gli utenti in oggetto
 - Utilizzare una metrica, per esempio, che riduca linearmente il peso quando il numero di elementi co-rated è bassa
- Amplificazione dei simili
 - Dare più peso agli utenti "molto simili", i.e. il cui il valore di similarità è vicino ad 1
- Selezione del numero di utenti simili
 - Utilizzare una soglia di similarità oppure un numero fisso di vicini



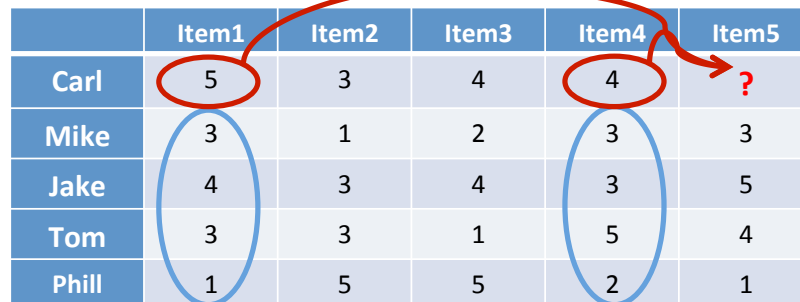
Approcci Memory e Model based

- Il CF *user-based* è considerato **basato sulla memoria**
 - la matrice di valutazione viene direttamente utilizzata per trovare gli utenti simili e per fare previsioni
 - può non scalare per la maggior parte degli scenari del mondo reale
- Approcci **model-based**
 - Basati su una fase di pre-elaborazione o *model learning*
 - In fase di esecuzione, solo il modello addestrato è usato per fare previsioni
 - I modelli sono aggiornati/ri-addestrati periodicamente
 - Grande varietà di tecniche utilizzate
 - La costruzione di modelli e l'aggiornamento può essere computazionalmente costosa
 - Vediamo ora il CF *Item-based*, un esempio di approccio model-based



Item-based Collaborative Filtering

- Idea di base e obiettivo
 - Utilizzare la similarità tra prodotti (e non utenti) per fare previsioni
 - Prevedere quale voto darebbe Carl all'item5
- Esempio:
 - individuare gli articoli che sono più simili ad item5 (sono item1 e item4)
 - Utilizzare i giudizi di Carl su item1 e item4 per predire quello per item5



	Item1	Item2	Item3	Item4	Item5
Carl	5	3	4	4	?
Mike	3	1	2	3	3
Jake	4	3	4	3	5
Tom	3	3	1	5	4
Phill	1	5	5	2	1

Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

25



Item-based Collaborative Filtering: Parametri

- Parametri simili all'approccio User-Based
- Tecniche per la misura della similarità:
 - In questo caso la letteratura dimostra che la similarità coseno funziona meglio
 - Una variante alla normale similarità coseno è quella di considerare la distanza di ogni giudizio dal valore medio di ogni utente
- La dimensione dell'insieme degli item simili per fare la previsione
 - è in genere fissa e minore dell'insieme completo dei simili
 - Un'analisi del dataset MovieLens indica che "nella maggior parte delle situazioni del mondo reale, un set di simili da 20 a 50 sembra ragionevole" [Herlocker et al. 2002]
- Una tecnica comune per predire la raccomandazione:

u = utente p = prodotto

$r_{u,i}$ = rate di u del prodotto i

$ratedItem(u)$ = prodotti valutati da u

$$pred(u, p) = \frac{\sum_{i \in ratedItem(u)} sim(i, p) \cdot r_{u,i}}{\sum_{i \in ratedItem(u)} sim(i, p)}$$



Pre-processing per l'approccio Item-based

- L'approccio Item-based non risolve il problema della scalabilità
- Approccio di pre-processing di Amazon.com (nel 2003)
 - Calcolare tutte le similarità tra coppie di item in anticipo
 - Il set dei simili da utilizzare in fase di esecuzione è in genere piuttosto piccolo, perché tiene conto solo dei prodotti che l'utente ha valutato
 - le similarità tra prodotti dovrebbero essere più stabili di quelle tra utenti
- Requisiti di memoria
 - Fino a N^2 coppie di similarità da memorizzare (N = numero di prodotti)
 - In pratica, questo valore è significativamente più basso (prodotti senza co-rating)
 - metodi per limitare l'occupazione di memoria
 - Soglia minima per co-rating, ossia num. minimo di utenti che hanno acquistato/valutato entrambi i prodotti
 - Limitare a priori la dimensione del set dei simili

Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

27



Cold Start e Sparsità dei Dati

- Problema di **cold start**
 - Come raccomandare nuovi elementi? Cosa consigliare ai nuovi utenti?
- Approcci diretti
 - Chiedere/forzare gli utenti di valutare una serie di prodotti
 - Utilizzare un metodo content-based oppure non personalizzato ossia basato sulla popolarità generale dei prodotti
 - oppure inizialmente semplicemente non raccomandare nulla
- Alternative
 - Utilizzare algoritmi specifici (non Nearest Neighbor)
 - Esempio:
 - Nell'approccio Nearest Neighbor, il set dei simili sufficientemente simili potrebbe essere troppo piccolo per fare buone previsioni
 - Assumere "transitività" dei simili, es. A simile a B, B a C, quindi A simile a C

Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

28



Esempio di Approccio a Dati Sparsi

- **Recursive Collaborative Filtering** [Zhang and Pu, 2007]
 - Se esiste un utente (*Mike*) molto simile all'utente considerato (*Carl*), che però anch'esso non ha votato per un prodotto (*Item5*), utilizzare CF per trovare il rating di *Mike* per *Item5*
 - Infine utilizzare *Mike* nel calcolo dei simili a *Carl*, invece che utilizzare utenti molto diversi a causa della sparsità dei dati

	Item1	Item2	Item3	Item4	Item5
Carl	5	3	4	4	?
Mike	5	2	4	3	?
Jake	4	3	4	3	5
Tom	3	3	1	5	4
Phill	1	5	5	2	1

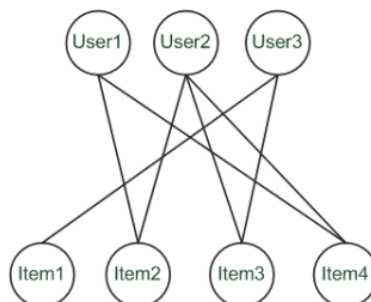
Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

29



Metodi Graph-based (I)

- **Spreading activation** [Huang et al. 2004]
 - Sfruttare la "transitività" presunta dei gusti del cliente e, quindi, aumentare la matrice con informazioni aggiuntive
 - Supponiamo di dover raccomandare un prodotto per *User1*
 - Con un approccio CF **standard**, *User2* sarà considerato un vicino di *User1* perché entrambi hanno acquistato *Item2* e *Item4*
 - *Item3* sarà raccomandato a *User1* considerando gli acquisti del vicino più prossimo (*User2*)
 - Vediamo ora che accade sfruttando più in profondità la transitività



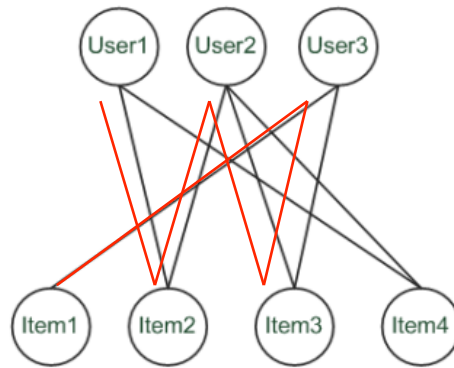
Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

30



Metodi Graph-based (II)

- Il valore standard della lunghezza del percorso in user o Item-based è 3
 - *Item3* è rilevante per *User1* perché esiste un percorso di tre passi (*User1* - *Item2* - *User2* - *Item3*) tra di loro
- tuttavia, poiché il numero di percorsi di lunghezza 3 può essere piccolo quando si trattano dati sparsi, l'idea per calcolare le raccomandazioni è quella di considerare anche percorsi più lunghi (ossia associazioni indirette)
 - ad esempio utilizzando percorsi a 5 passi verrebbe predetto anche *Item1*



Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

31



Altri metodi model-based

- Negli ultimi anni sono state proposte numerose tecniche
 - Tecniche statistiche di matrix factorization
 - singular value decomposition, principal component analysis
 - Regole associative
 - Utilizzata ad esempio per la market basket analysis
 - Modelli probabilistici
 - Modelli di clustering, reti bayesiane, Latent Semantic Analysis probabilistico
 - Approcci ad apprendimento automatico più complessi
- I costi di pre-processing
 - Di solito non discussi ma generalmente non sempre sostenibili
 - Aggiornamenti incrementali sono possibili ? è un tema di ricerca



Riduzione della Dimensionalità: SVD

- Proposta nel 2000 da B. Sarwar et al., WebKDD Workshop
- Idea di base: creare modelli più complessi offline per la produzione di previsioni più veloci on-line
- Singular Value Decomposition (SVD) per la riduzione della dimensionalità delle matrici di rating
 - genera un nuovo spazio con nuovi assi dove colloca i dati della matrice
 - i nuovi assi rappresentano le dimensioni lungo le quali i dati variano maggiormente, tuttavia non sono sempre facilmente interpretabili
 - cattura i segnali rilevanti nei dati filtrando il rumore con un numero k di dimensioni molto inferiore alle dimensioni originali ($20 \leq k \leq 100$)
- Raccomandazioni fornite in tempo costante
- Approccio popolare anche in Information Retrieval
 - Latent Semantic Indexing, compressione di dati, di immagini ...



Matrix Factorization SVD

- La **Singular Value Decomposition** (SVD) [Golub and Kahan, 1965] afferma che una matrice M può essere fattorizzata nel prodotto di 3 matrici:

$$M = U \times \Sigma \times V^T$$
 - Dove U contiene gli autovettori destri di MM^T (similarità tra utenti) e V gli autovettori sinistri di $M^T M$ (similarità tra prodotti)
 - gli autovettori sono ortonormali e costituiscono la base di un nuovo sistema di riferimento i cui assi sono le variabili latenti
 - I valori della matrice diagonale Σ sono gli autovalori, detti anche valori singolari, in ordine decrescente sulla diagonale
 - ogni autovalore è la varianza dei dati sulla dimensione che rappresenta
- La matrice di partenza si può approssimare utilizzando nel prodotto delle 3 matrici i primi k valori singolari maggiori in Σ



Esempio di predizione con SVD

M matrice di rating utenti-film

- SVD: $M_k = U_k \times \Sigma_k \times V_k^T$

U_k	Dim1	Dim2
Alice	0.47	-0.30
Bob	-0.44	0.23
Mary	0.70	-0.06
Sue	0.31	0.93

	Terminator	Die Hard	Twins	Eat Pray Love	Pretty Woman
V_k^T					
Dim1	-0.44	-0.57	0.06	0.38	0.57
Dim2	0.58	-0.66	0.26	0.18	-0.36

Σ_k	Dim1	Dim2
Dim1	5.63	0
Dim2	0	3.23

- rating previsto del prodotto i (Eat Pray Love) per l'utente u (Alice) con i primi 2 autovalori ($k=2$):

$$\hat{r}_{ui} = \bar{r}_u + U_k(Alice) \times \Sigma_k \times V_k^T(EPL)$$

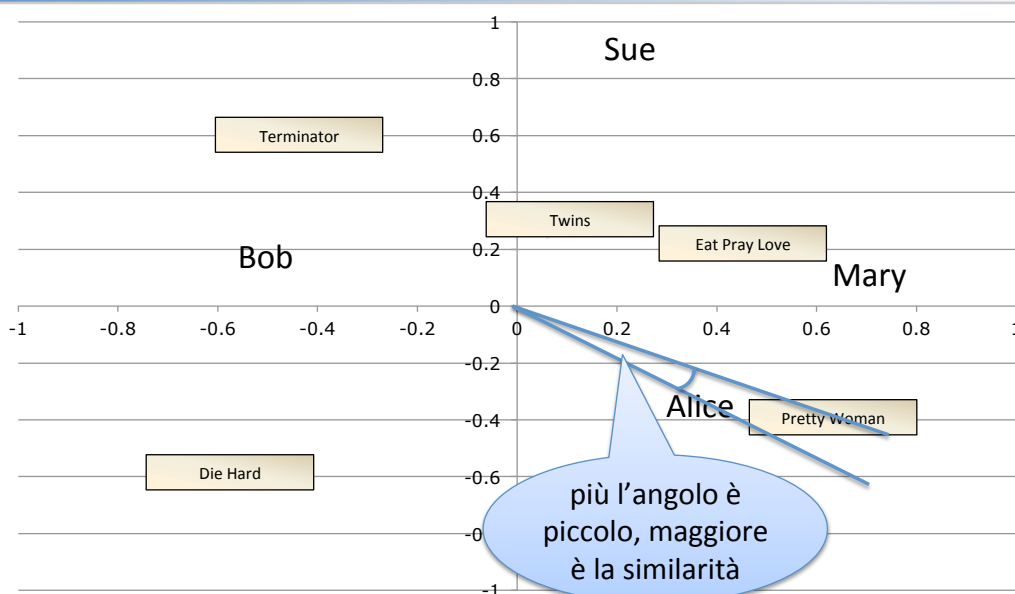
$$= 3 + (0.47 \times 5.63 \times 0.38 + -0.3 \times 3.23 \times 0.18) = 3.84$$

Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

35



Proiezione di U e V^T nello spazio bidimensionale ($k=2$)



Previsioni: e.g. **Alice** dovrebbe prediligere più **Pretty Woman** rispetto agli altri film, **Mary** invece **Eat Pray Love**, mentre **Sue** apprezzerà **Twins** (similitudine coseno)



Riduzione della dimensionalità con SVD: Conclusioni

- Matrix factorization SVD
 - Individuazione dei fattori latenti, ossia le dimensioni del nuovo spazio
 - Genera approssimazioni di matrici a basso rango k (num. autovalori scelti)
 - Proiezione di oggetti e utenti nello stesso spazio n -dimensionale
- la qualità della previsione dipende dalla giusta scelta di k , ossia dal numero dei valori singolari
 - I parametri possono essere determinati e ottimizzati solo su esperimenti in un determinato dominio
 - Koren et al. 2009 ha rilevato che di norma si ottengono risultati migliori con 20 - 100 fattori
 - perciò l'accuratezza delle recommendation può anche diminuire rispetto all'impiego della matrice di rating originale con valori di k inadeguati
- Versione efficiente in python: `randomized_svd()`

Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

37



Raccomandazioni con Regole Associative (i)

- Utilizzate comunemente per l'analisi impersonale dei comportamenti di acquisto
 - *“se l'utente acquista birra, acquista anche pannolini nel 70% dei casi”*
 - le regole associative si ricavano dall'analisi delle frequenze delle combinazioni di acquisto dall'intero insieme di transazioni di acquisto
- Le **regole associative** sono implicazioni statistiche del tipo $X \rightarrow Y$, es: birra \rightarrow pannolini (birra=**antecedente**, pannolini=**conseguente**)
 - in generale X e Y possono contenere più prodotti
- Misure di accuratezza delle regole: **supporto** e **confidenza**
 - il **supporto** di una regola è il numero di transazioni (di tutti gli utenti) dove X è acquistato insieme ad Y , diviso il num. totale di transazioni
 - la **confidenza** ha lo stesso numeratore del supporto, mentre il denominatore è il num. delle transazioni che contengono X
 - Utilizzati anche come cut-off per scegliere le regole migliori

Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

38



Raccomandazioni con Regole Associative (ii)

- Approccio più semplice
 - Trasformare rating (es 1-5) in binario
 - Voti: 1 = se sopra la media utente
- Esempio di regola associativa:
 - Item1 → item5 escluso Carl
 - supporto (2/4), confidenza (2/2)
- Formulare raccomandazioni per Carl (metodo di base)
 - Estrarre le regole associative dal set di transazioni
 - Determinare i prodotti non acquistati da Carl, es. item5
 - Determinare le regole *rilevanti* per Carl in base alle proprie transazioni, ad esempio la regola item1->item5 è rilevante per Carl perché ha acquistato item1 ma non item5
 - Ordinare le recommendation in base alla confidenza delle regole rilevanti e proporre il prodotto **conseguente** della prima regola

	Item 1	Item 2	Item 3	Item 4	Item 5
Carl	1	0	0	0	?
Mike	1	0	1	0	1
Jake	1	0	1	0	1
Tom	0	0	0	1	1
Phill	0	1	1	0	0

Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

39



Metodi Probabilistici

- Idea di base:
 - qual è la probabilità che un utente esprima un rating r di un prodotto p dati i rating dell'utente e di tutti gli altri utenti ?
 - sia Y la variabile aleatoria relativa al rating su p ed X i rating dell'utente
 - quindi occorre calcolare $P(Y, X) = P(Y|X) P(X) = P(X|Y)P(Y)$ (bayes)
 - assunzione: i rating X_i sui prodotti d sono considerati indipendenti tra loro

$$P(Y|X) = \frac{P(X|Y) \times P(Y)}{P(X)}$$

$$P(Y|X) = \frac{\prod_{i=1}^d P(X_i|Y) \times P(Y)}{P(X)}$$

dato che $P(X)$ è invariante rispetto ad Y è sufficiente

determinare quel Y che massimizza $\operatorname{argmax}_Y \prod_{i=1}^d P(X_i | Y) \cdot P(Y)$

- Esistono tecniche in letteratura ancora più complesse
 - Utilizzo di clustering, Bayesian Networks, pLSA ...



Metodi Probabilistici: Esempio

Valutazioni di Carl

$X = [Item1=1, Item2=3, Item3=3, Item4=2]$

Qual è la valutazione più probabile di Carl per item5 ?

	Item1	Item2	Item3	Item4	Item5
Carl	1	3	3	2	?
Mike	2	4	2	2	4
Jake	1	3	3	5	1
Tom	4	5	2	3	3
Phill	1	1	5	2	1

occorre determinare qual è il **valore** di *item5* che massimizza la probabilità condizionata $P(X=[Item1=1, Item2=3, Item3=3, Item4=2] \mid Item5 = \text{valore})$ perciò occorre calcolare la probabilità per ogni valore possibile di *item5*

$$P(X \mid Item5=1) = P(Item1=1 \mid Item5=1) \times P(Item2=3 \mid Item5=1) \times P(Item3=3 \mid Item5=1) \times P(Item4=1 \mid Item5=1) \times P(Item5=1) \\ = 2/2 \times 1/2 \times 1/2 \times 1/2 \times 2/4 = 0.062$$

$$P(X \mid Item5=2) = 0 \times 0 \times 0 \times 0 \times 0 = 0 = P(X \mid Item5=5)$$

$$P(X \mid Item5=3) = 0 \times 0 \times 0 \times 0 \times 1/4 = 0$$

$$P(X \mid Item5=4) = 0 \times 0 \times 0 \times 0 \times 1/4 = 0$$

se per un dato Y_i in $P(X \mid Y_i)$, nessun utente ha dato ad un prodotto X_j lo stesso voto di Carl, la prob. $P(X_j \mid Y_i)$ si azzerava azzerando l'intera $P(X \mid Y_i)$
Risolto con metodi di smoothing

Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

41



Slope One Predictor (I)

- Proposto da Lemire and Maclachlan (2005)
- Idea: Il predittore **Slope One** è semplice e si basa su un differenziale di popolarità tra gli elementi per gli utenti

– Esempio:

	Item1	Item5
Carl	2	?
Jack	1	3

$$p(\text{Carl}, \text{item5}) = 2 + (3 - 1) = 4$$

- Schema di base: prendere la media di queste **differenze dei co-rating** per fare la previsione
- In generale: trovare una funzione della forma $f(x) = x + b$
 - Ecco perché il nome è "Slope One"

Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

42



Slope One Predictor (II)

- Il caso precedente mostra il calcolo di una predizione considerando una coppia di utenti e una coppia di prodotti
- La generalizzazione del metodo avviene creando una media delle differenze con più utenti
- Nell'esempio in tabella a lato, la differenza media dei punteggi tra l'item **B** e **A** è $((5-3) + (3-4)) / 2 = 0,5$
 - basandosi sui voti di Carl e Jack, che hanno votato entrambi gli Item
 - Allo stesso modo, la differenza media tra l'Item **C** e **A** è $(5-2)/1=3$
- Per prevedere il voto di Lucy sull'Item **A** rispetto ai suoi voti su **B** e **C**
 - usiamo il suo voto per l'item **B** ottenendo $2 + 0,5 = 2,5$
 - analogamente usiamo il suo voto per l'item **C** ottenendo $5 + 3 = 8$
 - segue nella prossima slide come combinare i due risultati

	Item A	Item B	Item C
Carl	5	3	2
Jack	3	4	?
Lucy	?	2	5



Slope One Predictor (III)

- Se un utente ha votato diversi elementi, le previsioni possono essere combinate utilizzando una media ponderata
 - una buona scelta per il peso è il numero di utenti che hanno valutato entrambi gli elementi
- Nell'esempio precedente, si può quindi prevedere il voto di Lucy per l'Item A come:

$$\frac{2 \times 2.5 + 1 \times 8}{2 + 1} = \frac{13}{3} = 4.33$$

voto predetto di
Lucy sull'item A

Peso: 2 utenti hanno
votato sia B che A

Peso: 1 utente ha votato
sia C che A



Riassunto sui metodi Collaborative Filtering

- Pro:
 - Facile da comprendere e implementare, funziona bene in alcuni settori
- Contro:
 - richiede comunità di utenti, problemi di sparsità, nessuna integrazione di altre fonti di conoscenza
- Qual è il miglior metodo di CF ?
 - Difficile da dirsi, differenze tra i metodi sono spesso molto piccole ($\approx 1\%$)
- Come valutare l'accuratezza di previsione ?
 - MAE / RMSE: quale metodo è migliore per valutare un RS ?
 - Nonostante entrambi abbiano pro e contro. In generale il metodo più utilizzato è il RMSE
 - Serendipity (novità e sorpresa generate dalle raccomandazioni)
 - Non ancora utilizzato in modo esteso

Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

45



il Caso Google News

- Aggrega notizie provenienti da diverse migliaia di fonti
- Le visualizza agli utenti loggati in modo personalizzato
- Raccomandazione basata su un approccio collaborativo
 - Sulla base della storia dei click dell'utente attivo e della storia della comunità
- Principali sfide:
 - Vasto numero di articoli e utenti
 - Generare lista di recommendation in tempo reale (al massimo un secondo)
 - Flusso costante di nuovi elementi
- Sono necessari sforzi significativi per quanto riguarda gli algoritmi, l'ingegneria e la parallelizzazione

Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

46



Google News Personalization Engine

Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

47



Google News

- Gli approcci memory-based puri non sono direttamente applicabili
- con gli approcci model-based c'è il problema di aggiornamento continuo del modello
 - Viene utilizzata una combinazione di tecniche model and memory-based
- Parte model-based: Vengono utilizzate 2 tecniche di clustering
 - Latent Semantic Indexing Probabilistica (pLSI) come proposto da (Hofmann 2004)
 - MinHash, metodo di hashing per stimare la similarità di due insiemi
- Parte memory-based: trattare i nuovi utenti analizzando lo storico delle *co-visite* con altri utenti
- **MapReduce** (di Google) è utilizzata per parallelizzare la computazione e renderla scalabile

Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

48



Letteratura (I)

- [Adomavicius and Tuzhilin 2005] Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, *IEEE Transactions on Knowledge and Data Engineering* 17 (2005), no. 6, 734–749
- [Breese et al. 1998] Empirical analysis of predictive algorithms for collaborative filtering, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence* (Madison, WI) (Gregory F. Cooper and Serafín Moral, eds.), Morgan Kaufmann, 1998, pp. 43–52
- [Gedikli et al. 2011] RF-Rec: Fast and accurate computation of recommendations based on rating frequencies, *Proceedings of the 13th IEEE Conference on Commerce and Enterprise Computing - CEC 2011, Luxembourg, 2011, forthcoming*
- [Goldberg et al. 2001] Eigentaste: A constant time collaborative filtering algorithm, *Information Retrieval* 4 (2001), no. 2, 133–151
- [Golub and Kahan 1965] Calculating the singular values and pseudo-inverse of a matrix, *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis* 2 (1965), no. 2, 205–224
- [Herlocker et al. 2002] An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms, *Information Retrieval* 5 (2002), no. 4, 287–310
- [Herlocker et al. 2004] Evaluating collaborative filtering recommender systems, *ACM Transactions on Information Systems (TOIS)* 22 (2004), no. 1, 5–53



Letteratura (II)

- [Hofmann 2004] Latent semantic models for collaborative filtering, *ACM Transactions on Information Systems* 22 (2004), no. 1, 89–115
- [Huang et al. 2004] Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering, *ACM Transactions on Information Systems* 22 (2004), no. 1, 116–142
- [Koren et al. 2009] *Matrix factorization techniques for recommender systems*, *Computer* 42 (2009), no. 8, 30–37
- [Lemire and Maclachlan 2005] Slope one predictors for online rating-based collaborative filtering, *Proceedings of the 5th SIAM International Conference on Data Mining (SDM '05)* (Newport Beach, CA), 2005, pp. 471–480
- [Sarwar et al. 2000a] Application of dimensionality reduction in recommender systems – a case study, *Proceedings of the ACM WebKDD Workshop* (Boston), 2000
- [Zhang and Pu 2007] A recursive prediction algorithm for collaborative filtering recommender systems, *Proceedings of the 2007 ACM Conference on Recommender Systems (RecSys '07)* (Minneapolis, MN), ACM, 2007, pp. 57–64



Recommendation in Python: **Surprise**

- **Surprise** è una libreria Python per la costruzione e la validazione di sistemi di recommendation
- I dati per costruire i modelli possono essere caricati da file CSV o da DataFrame pandas
 - è anche possibile reperire e caricare dataset d'esempio con una singola istruzione
- Fornisce diversi algoritmi di recommendation
 - user-based, item-based, scomposizione di matrici, ...
 - diverse misure di similarità: coseno, Pearson, ...
- Fornisce strumenti per la validazione dei modelli ottenuti
 - split train-test, cross-validation, ...

```
>>> import surprise
```



Surprise: Dati d'Esempio

- La classe `Dataset` fornisce diversi metodi per caricare dati da utilizzare per la costruzione e validazione di modelli
- La funzione `load_builtin` carica uno di alcuni set di dati usati tipicamente per testare modelli di recommendation
 - ogni dataset è scaricato automaticamente se necessario
 - i dataset sono salvati di default nella directory `".surprise_data"`

```
>>> data = surprise.Dataset.load_builtin("ml-100k")
```

- Tra i dataset disponibili ci sono:
 - "ml-100k": dataset benchmark per recommendation di film, contiene 100.000 rating di 1.000 utenti su 1.700 film
 - "ml-1m": come sopra con 1 milione di rating, 6.000 utenti, 4.000 film
 - "jester": dati dal sito Jester (<http://eigentaste.berkeley.edu/>) che consiglia barzellette agli utenti; circa 1,8 mln. rating di 60.000 utenti



Surprise: Caricamento da CSV

- Con la funzione `load_from_csv` si caricano dati da file CSV
 - Alla funzione deve essere passato un oggetto `Reader` che specifichi alcune opzioni
 - `line_format`: ordine delle colonne, di default user item rating (ID utente, ID prodotto, voto assegnato)
 - `sep`: separatore di colonna nel file
 - `rating_scale`: tupla con valori minimo e massimo per il voto
- ```
reader = surprise.Reader(rating_scale=(1, 5))
data = surprise.Dataset.load_from_csv(
 "ratings.csv", reader)
```



## Surprise: Caricamento Dati Esterni

- Con `load_from_df` si caricano dati da un `DataFrame` pandas
  - il `DataFrame` può a sua volta essere caricato da molte fonti diverse (CSV, database SQL, JSON, ...)
  - deve avere tre colonne: ID utente, ID prodotto, punteggio
  - nel `Reader` è sufficiente dichiarare la `rating_scale`

*# esempio: caricare dati da PostgreSQL*

```
import psycopg2; import pandas as pd
with psycopg2.connect(...) as conn:
 df = pd.read_sql("SELECT ... ", conn)
reader = surprise.Reader(rating_scale=(1, 5))
data = surprise.Dataset.load_from_df(df, reader)
```



## Surprise: Dataset e Trainset

- Un **Dataset** di Surprise memorizza i dati con cui è stato costruito in **forma grezza**
    - utenti e prodotti sono identificati dagli **oggetti usati nei dati originali**, detti ID *raw* (“grezzi”), che possono essere numeri, stringhe, ...
  - Da un **Dataset** può essere estratto un **Trainset**, che memorizza i dati in **forma ottimizzata**
    - agli  $N$  utenti distinti del **Dataset** sono assegnati ID “interni” (*inner*), numeri interi da 0 a  $N-1$ ; lo stesso avviene per i prodotti
  - Per costruire un **Trainset** da un intero **Dataset**, utilizzarne il metodo **build\_full\_trainset**
    - nella pratica si utilizzerà solo una parte di **Dataset**, come vedremo...
- ```
>>> full_trainset = data.build_full_trainset()
```



Surprise: Attributi e Metodi di Trainset

- **n_users**, **n_items**, **n_ratings**: numero complessivo di utenti, prodotti e voti presenti nel dataset
- **all_users()**, **all_items()**: iteratori di tutti gli ID di utenti e prodotti, equivalenti a **range(n_users o n_items)**
- **ur**: dizionario che associa ad ogni ID utente una lista dei voti dati come tuple (**id_prodotto**, **voto**)
- **ir**: dizionario da ID prodotti a lista tuple (**id_utente**, **voto**)
- **all_ratings()**: iteratore di tutti i voti in forma di tuple (**id_utente**, **id_prodotto**, **voto**)
- **to_inner_uid(x)**, **to_inner_iid(x)**: converte un ID *grezzo* x di un utente o prodotto in un ID *interno*
- **to_raw_uid(x)**, **to_raw_iid(x)**: operazioni inverse
- **global_mean**: media di tutti i voti



Surprise: Addestrare un Modello

- Surprise permette di addestrare modelli di recommendation utilizzando diversi algoritmi
- Dato un algoritmo da usare si inizia creando un modello “vuoto” basato su di esso, fornendo eventuali parametri
- Ad esempio, per definire un semplice user-based recommender:

```
>>> model = surprise.KNNBasic()
```

- Per addestrare il modello definito, va quindi usato il metodo `fit` passando il `Trainset` su cui addestrarsi

```
>>> model.fit(full_trainset)
```



Surprise: Prevedere un Voto dal Modello

- Col metodo `predict` è possibile ottenere il voto previsto dal modello per un oggetto da parte di un utente
- Restituisce una `Prediction`, simile ad una tupla con attributi:
 - `uid`: ID raw dell'utente (primo argomento di `predict`)
 - `iid`: ID raw dell'oggetto (secondo argomento di `predict`)
 - `r_ui`: voto reale (terzo argomento di `predict`, opzionale)
 - questo dato è usato per il calcolo dell'accuratezza
 - `est`: voto predetto
 - `details`: dizionario con informazioni aggiuntive



Surprise: Prevedere Molteplici Voti

- Da un `Trainset`, è possibile ottenere un *test set* di coppie utente-oggetto per cui non c'è un voto noto

```
>>> full_testset =  
full_trainset.build_anti_testset()
```

- Si ha una lista di tuple (*id_utente*, *id_oggetto*, *voto*), dove gli ID sono quelli raw e il voto è la media dei voti noti
- Tramite il metodo `test` del modello, si ottengono i voti predetti corrispondenti

```
>>> preds = model.test(full_testset)
```

- In questo modo otteniamo le stime di tutti i voti non noti



Surprise: Ottenere gli *N* Migliori Suggerimenti per ogni Utente

- Possiamo scrivere una funzione che ottenga i migliori suggerimenti per ogni utente dalle predizioni così estratte

```
from collections import defaultdict  
def get_top_recommendations(predictions, topN = 3):  
    top_recs = defaultdict(list)  
    # raccogli liste di suggerimenti per utente  
    for uid, iid, true_r, est, _ in predictions:  
        top_recs[uid].append((iid, est))  
    # ordina ogni lista per voto e prendi i primi N  
    for uid, user_ratings in top_recs.items():  
        user_ratings.sort(key = lambda x: x[1],  
                          reverse = True)  
        top_recs[uid] = user_ratings[:topN]  
    return top_recs
```



Surprise: Modelli User-based e Item-based

- La classe `KNNBasic` implementa un semplice algoritmo di collaborative filtering basato su similarità tra utenti o oggetti
 - il voto previsto per una coppia utente-oggetto è la media pesata sulla similarità dei voti dati dagli utenti simili o per gli oggetti simili
- I parametri impostabili sono:
 - `k`: numero massimo di vicini da considerare (default 40)
 - `min_k`: numero minimo di vicini (default 1)
 - se non raggiunto il voto previsto è la media globale dei voti noti
 - `sim_options`: configurazione della misura di similarità
- A sua volta in `sim_options` si può indicare
 - `name`: nome della misura di similarità da usare
 - `user_based`: `True` (default) o `False` (item-based)
 - `min_support`: num. minimo di oggetti tra utenti simili (o viceversa)

Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

61



Surprise: Modelli con Scomposizione di Matrici

- Surprise offre diversi modelli basati su scomposizione di matrici, addestrati tramite discesa del gradiente *stocastica*
 - ad ogni passaggio d'addestramento il *gradiente dell'errore è calcolato* non sull'intero training set ma *su un singolo elemento*
- L'algoritmo più semplice di questo tipo è SVD


```
svd = surprise.prediction_algorithms. \
    matrix_factorization.SVD(n_factors=100)
```
- Tra gli attributi impostabili ci sono:
 - `n_factors`: numero di fattori in cui scomporre i dati
 - `n_epochs`: durata dell'addestramento in *epoche*, in ciascuna ogni elemento del training set è utilizzato una volta, in ordine casuale
 - `lr_all`: lunghezza del passo (*learning rate*) per la discesa



Surprise: Funzionamento di SVD

- Siano U , I e F il numero di utenti, di oggetti e di fattori
- Tramite l'addestramento vengono individuati i seguenti array, accessibili come attributi del modello
 - una matrice $U \times F$ p_u con le rappresentazioni di ciascun utente come vettore degli F fattori
 - una matrice $I \times F$ q_i con simili rappresentazioni per gli oggetti
 - i vettori b_u e b_i con i *bias* per ciascun utente e oggetto, che indicano quanto i voti di ciascuno si discostino in genere dalla media
- Da questi il voto previsto per utente u ed oggetto i è dato da

media globale dei voti

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

somma per ogni fattore del prodotto tra il suo peso per l'utente e per l'oggetto

- ... o come si scrive in NumPy:

```
global_mean + bu[u] + bi[i] + qi[i].dot(pu[u])
```



Surprise: Validazione dei Modelli

- Surprise offre diverse funzioni per valutare l'accuratezza dei modelli di recommendation generati
 - suddivisione dei dataset di rating in training e test set
 - calcolo di diverse misure di accuratezza, tra cui RMSE
- Un test set è una lista di tuple (*ID utente, ID oggetto, voto reale*) non presente nel rispettivo training set
- Tramite il metodo test di un modello, passando una tale lista di tuple, otteniamo la lista di oggetti Prediction corrispondenti
 - ciascun oggetto incapsula voto predetto e voto reale
- Da tale lista si possono calcolare diverse metriche di accuratezza



Surprise: Validazione con Hold-out

- Per dividere un `Dataset` in training e test set, Surprise offre una funzione `train_test_split` simile a quella di scikit-learn

```
>>> trainset, testset = surprise.model_selection. \
    train_test_split(data, test_size=0.3)
```

- Effettuata la divisione, possiamo addestrare un modello sul training set...

```
>>> model = surprise.KNNBasic()
```

```
>>> model.fit(trainset)
```

- ...ed ottenere i voti predetti sul test set

```
>>> preds = model.test(testset)
```

- Da queste predizioni è possibile ad es. calcolare l'RMSE

```
>>> surprise.accuracy.rmse(preds)
0.9822
```

Data Intensive Applications - G. Moro, R. Pasolini - DISI, Università di Bologna, Cesena

65



Surprise: Cross Validation

- La *N-fold cross validation* costituisce un'alternativa al metodo hold-out per la valutazione di un modello
 - i rating sono suddivisi in N insiemi (*fold*) di pari dimensioni
 - ciascun fold viene usato come test set di un modello addestrato sull'unione di tutti gli altri, ottenendo così N valori di accuratezza
 - come accuratezza finale si considera la media degli N valori
- Surprise offre funzionalità apposite per generare i fold, utilizzabili per eseguire manualmente la valutazione
- In alternativa viene offerto un metodo `cross_validate` che esegue automaticamente tutto il procedimento
 - vanno specificati l'algoritmo da usare, il dataset e la lista dei nomi delle misure da generare



Surprise: Esempio di Cross Validation

```
# carico il dataset
data = surprise.Dataset.load_builtin('ml-100k')
# definisco algoritmo da usare e eventuali parametri
algo = surprise.KNNBasic()
# eseguo la cross validation a 5 fold
cross_validate(algo, data, cv=5, verbose=True)
```

- Altri parametri specificabili in `cross_validate` sono:
 - `measures`: lista con nomi delle misure di accuratezza da estrarre (default RMSE e MAE)
 - `n_jobs`: numero di core CPU da usare in parallelo (default 1)

