Frequent Itemset Mining & Association Rules

1. Market Basket Model

- Nasce per analizzare le associazioni tra acquisti nei supermercati
- Dato che alcuni prodotti sono spesso comprati insieme, possiamo individuare itemset frequenti e generare regole di associazione
- **Obiettivo**
  - Identificare oggetti che vengono comprati insieme da clienti sufficientemente numerosi
- **Approccio**
  - Trovare dipendenze tra gli oggetti
- **Esempio regola**

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

  - Da qui vediamo che se un apersona prende pannolini e/o latte, probabilmente prende anche la birra
- Un set di item che compara in diversi "basket" è detto frequente
  - Sia
    - s una soglia di supporto
      - Rappresenta il numero minimo di volte che un set di articoli deve apparire nei carrelli per essere considerato "frequente"
    - I è un itemset
      - Il supporto per I è il numero di quante volte I compare come sottoinsieme in un carrello
        - Ex. Ho i carrelli
          - Carrello 1: {latte, pane, uova}
          - Carrello 2: {pane, burro}
          - Carrello 3: {latte, pane, biscotti}
          - Carrello 4: {latte, uova}
        - Il supporto per l'itemset I = {latte, pane} è 2
          - Compare nel Carrello 1 e nel Carrello 3
  - I è un frequent itemset se il suo supporto è maggiore di S
- **Esempio**
  - Items = {milk, cola, peas, beer, juice}
  - s = 3
    - Ergo considero frequenti itemsets che compaiono per 3 o più volte
  - B1 = {m, c, b} B2 = {m, p, j}
    B3 = {m, b} B4= {c, j}
    B5 = {m, p, b} B6 = {m, c, b, j}
    B7 = {c, b, j} B8 = {b, c}
    Dati questi carrelli
  - I frequent itemsets saranno
    - {m}
      - Compare in 5 carrelli
    - {c}
      - Compare in 5 carrelli
    - {b}
      - Compare in 6 carrelli
    - {j}
      - Compare 3 volte
    - {m, b}
      - Compaiono insieme per 3 volte
    - {b, c}
      - Compaiono insieme per 3 volte
    - {c, j}
      - Compaiono insieme per 3 volte

2. Metriche principali

- **Regole di associazione**
  - Una regola di associazione ha la forma I → j
    - I è un sottoinsieme di item
    - j un item
  - È interpretata come: se I è presente, allora j è probabile che lo sia
  - Ex. Se {latte, biscotti} è frequente, possiamo ottenere
    - latte → biscotti
    - biscotti → latte
- Le metriche principali utilizzate per valutare la bontà delle regole di associazione sono
  - **Supporto**
    - (sup(I))
    - Indica la proporzione di transazioni che contengono un dato temset I
    - È utilizzato per identificare gli itemset che compaiono frequentemente nel dataset
    - Ad esempio, se I compare in 4 transazioni su 10, il supporto sarà sup(I) = 0.4
  - **Confidenza**
    - (conf (I → j))
    - Rappresenta la probabilità che l'item j sia presente in una transazione dato che tutti gli item in I sono presenti
      - $\text{conf}(I \to j) = \frac{\text{support}(I \cup j)}{\text{support}(I)}$
    - La confidenza misura la forza dell'implicazione tra I e j
      - Più è vicina a 1, più l'associazione è forte
    - Non tutte le regole ad alta confidenza sono interessanti
      - Ex. Una regola che mette in relazione qualcosa e il latte non è detto che sia utile
        - Potrebbe essere solo che il latte è di per sé un prodotto acquistato molto spesso
      - Necessitiamo dunque anche della metrica di Interesse
  - **Interesse**
    - O lift improvement
    - Misura quanto la confidenza della regola I → j si discosta dalla semplice probabilità che j compaia da solo
      - $interesse(I \to j) = |conf(I \to j) - sup(j)|$
    - Se l'interesse è vicino a 0 significa che I non influenza significativamente la presenza di j
      - La regola è poco utile
    - Se l'interesse è alto allora potrebbe essere che I influenzi la presenza di j
- **Approccio generale per la ricerca di una regola**
  - Trovo gli itemset frequenti (con supporto ≥ s)
  - Da ciascun itemset, genero regole di associazione e calcolo la confidenza
  - Se la confidenza supera la soglia prefissata, la regola viene mantenuta
    - Eventualmente controllo anche l'interesse

3. Rappresentazione dei dati di Market-Basket

- **Problema della memoria**
  - Tipicamente i dati market basket sono memorizzati in un file basket per basket
  - Se la dimensione del file è grande
    - Non entra in memoria
  - Un costo importante per qualsiasi algoritmo è il costo di lettura dei basket dal file
  - Una volta che un blocco del disco pieno di baskets si trova in main memory possiamo espanderlo, generando tutti i possibili subset di dimensione k
    - $\binom{n}{k}$
      - Il tempo per generare tutti i sottoinsiemi di dimensione k per un basket con n articoli potrebbe dominare il tempo necessario per trasferire i dati da disco
        - Soprattutto per k grandi
      - Tipicamente abbiamo bisogno solo di piccoli itemset frequenti
        - K = 2 o K = 3
  - Quando abbiamo le coppie, il loro calcolo è un operazione che non è considerata ai fini delle performance
  - Ulteriore ottimizzazione
    - Se un item è sempre presente in tutti gli itemset, questo viene rimosso
    - Se esiste un elemento singolo frequente, quello formerà una coppia con
  - Normalmente per valutare l'onerosità di un algoritmo, che fa il calcolo dei frequent items, si conta quante volte è costretto a leggere il file complessivo dei basket
- **Altro problema - Itemset counting**
  - **Obiettivo**
    - Tutti gli algoritmi per gli insiemi di elementi frequenti richiedono il mantenimento di diversi conteggi durante il passaggio dei dati
      - Dobbiamo contare il numero di volte che ogni coppia di elementi appare nei basket
    - Se non abbiamo abbastanza memoria principale per memorizzare tutti i conteggi
      - Ogni operazione richiederà il caricamento di una pagina da disco
    - L'algoritmo andrà lentissimo se ogni conteggio si trova nella memoria principale
    - Ottimizzare gli algoritmi per usare al meglio la memoria
  - **Metodi**
    - **Matrice triangolare**
      - Usa un vettore V [k] per memorizzare la matrice [i, j] compatta
      - Risparmio di memoria (non ho un array in due dimensioni) ma serve una formula per l'indice k
        - $k = (i-1)\left(n - \frac{i}{2}\right) + j - i$
          - K
            - Posizione dell'elemento nel vettore
          - n
            - Numero di prodotti
          - (i, j)
            - Coppia di indici da mappare nel vettore
            - Attenzione!
              - i < j
              - Altrimenti non funziona
      - Invece che n^2 elementi ne ho
        - $\frac{n(n-1)}{2}$
      - **Vantaggio**
        - Memoria compatta
      - **Svantaggio**
        - Complessità nel calcolo dell'indice
      - Esempio
        - Ho 3 prodotti
          - Mela
          - Pera
          - Fragola
        - Vogliamo contare quante volte ogni coppia di prodotti compare insieme nei basket
          - Normalmente useremmo una matrice 3x3

	mela	pera	fragola
mela	X		
pera		X	
fragola			X

          - Notiamo però che
            - La diagonale non ci serve
              - (che ce ne facciamo della coppia mela - mela)
            - Gli indici (1, 2) e (2, 1) emetterebbero lo stesso valore
              - Quindi in realtà ci serve o la triangolare bassa o la triangolare alta senza diagonale
        - L'indice della coppia mela - pera è (1, 2) o (2, 1)
          - Usiamo (1, 2) così i < j
            - k = (1 - 1) (3 - (1/2)) + 2 - 1
              - 1
        - Usando la formula di prima
          - Coppia mela-fragola (1, 3)
            - k = (1 - 1) (3 - (1/2)) + 3 - 1
              - 2
          - Coppia pera - fragola (2, 3)
            - k = (2 - 1) (3 - (2/2)) + 3 - 2
              - 3
        - Ipotizzando che ci siano
          - 5 coppie (mela, pera)
          - 7 coppie (mela. fragola)
          - 9 copppie (pera, fragola)
        - Invece che salvarmi il matricione di 9 elementi in cui il triangolo basso e il triangolo alto sono uguali e la diagonale è inutile
          - Mi salvo un vettore di 3 elementi
            - [5, 7, 9]
          - Uso lo spazio usato da
            - $\frac{\binom{m}{2}}{\text{elementi}}$
              - m! / 2
                - 6/2 = 3
          - Piuttosto che m^2
            - 9
    - **Metodo delle triplette**
      - Conteggi memorizzati come triple [i, j, c]
      - Il conteggio della coppia (i,j) è c
      - Si utilizza una struttura dati, come una tabella hash con i e j come chiave di ricerca
        - In modo da poter dire se esiste una tripla per un dato i e j e, in caso affermativo, trovarla
      - Non richiede la memorizzazione di nulla se il conteggio di una coppia è 0
      - Il metodo delle triplette richiede di memorizzare tre numeri interi, anziché uno, per ogni coppia che appare in qualche basket
  - **Conclusioni**
    - La matrice triangolare è migliore se almeno 1/3 delle coppie possibili appare effettivamente in qualche basket
    - Il metodo delle triple è migliore se appare un numero significativamente inferiore a 1/3 delle coppie possibili
      - Ovvero è avvantaggiata se abbiamo dei buchi
- **Monotonicità**
  - Se un insieme I di elementi è frequente, allora lo è anche ogni sottoinsieme di I
  - Se viene data una soglia di supporto s
    - Diciamo che un insieme di elementi è **massimo** se nessun soprainsieme è frequente
  - Se elenchiamo solo gli insiemi di elementi massimi, allora sappiamo che
    - Tutti i sottoinsiemi di un insieme di elementi massimo sono frequenti
  - Alla base di queste ottimizzazioni c'è comunque l'utilizzo inverso della regola di monotonia
    - Solitamente, se (a,b,c) frequente, allora lo è anche (a,b)
    - Analogamente, e molto più importante
      - Se (a) non è frequente
        - Non lo sarà (a, b)
        - Non lo sarà nessun itemset contenente (a)
- Perché è così importante contare le coppie di item?
  - Solitamente le coppie sono meno dei terzetti possibili, difatti la regola generale sarebbe
    - $\binom{m}{2} \ll \binom{m}{3}$
  - Tuttavia in questo dominio è esattamente il contrario
    - $\binom{m}{2} \geq \binom{m}{3}$
      - Se una coppia non è un frequente, non lo sarà nemmeno il terzetto che contiene elementi che in coppia non sono frequenti
  - Da questo si capisce come l'obiettivo principale sia trovare le **coppie frequenti**

Presented with xmind