CS699
Lecture 7
Intervention
Association Rule Mining
Collaborative Filtering

Intervention

- We will briefly discuss following two intervention methods:
  - A/B test
  - Uplifting

A/B Test

- Randomized experiment for testing the causal effect of a treatment or intervention on outcomes of interest.
- Frequently used by marketing companies and companies who use internet platforms, such as Amazon, Google, Microsoft, and Uber, to test new features.
- Drug companies also use to compare different treatments.
- Will illustrate using a simple example.

A/B Test

- Example: Test new button color on a webpage.
- Old: blue, New: green
- Randomly sample 300 users and randomly split them into two groups, Group A and Group B, each with 150 users.
- Present old webpage to Group A and new webpage to Group B.
- Count how many users in each group clicked the button.
- Assume the following results:
  - Group A: 124 users clicked
  - Group B: 138 users clicked

A/B Test

- Summary

	Group A	Group B
Total # users	150	150
Click count	124	138
Click rate	0.827	0.92

- Analysis

$p_A = 0.827$, $p_B = 0.92$, $n_A = 150$, $n_B = 150$

$$SE = \sqrt{\frac{p_A(1-p_A)}{n_A} + \frac{p_B(1-p_B)}{n_B}} = 0.038$$

$$t\ statistic = \frac{p_B - p_A}{SE} = \frac{0.92-0.827}{0.038} = 2.447$$

P-value = 0.007; α = 0.05; reject null hypothesis

Difference is statistically significant => adopt new design

Null hypothesis H_0: $\mu_A = \mu_B$ (there is no statistically significant difference between two click rates)

Alternative hypothesis H_a: $\mu_A < \mu_B$ (the increase in click rate is statically significant)

Significance level α = 5%

Uplifting

- Consider you want to send a promotion to your customers.
- You can send the promotion to all customers.
- Or, you can send the promotions to only those customers who is more likely to purchase your product only if they receive the promotion.
- You can use “uplifting” to identify such customers.

- We illustrate basic concepts using a small example.
- L8.R code has a larger example.

Uplifting

- Consider a dataset with the following information about 10 consumers:

Income	Married	Age	Housing
high	yes	old	own
high	yes	young	rent
high	no	old	rent
middle	yes	young	own
high	yes	old	rent
high	yes	young	rent
middle	yes	old	rent
high	no	young	own
high	no	old	own
high	no	young	rent

Uplifting

- Split D into two groups of equal size, *treatment group* (group A) and *control group* (group B). Send promotions to group A and do not send promotions to group B.

Income	Married	Age	Housing	Promotion
high	yes	old	own	yes
high	yes	young	rent	yes
high	no	old	rent	yes
middle	yes	young	own	yes
high	yes	old	rent	yes
high	yes	young	rent	no
middle	yes	old	rent	no
high	no	young	own	no
high	no	old	own	no
high	no	young	rent	no

Uplifting

- Later, conduct a survey and record which consumers made purchase.

Income	Married	Age	Housing	Promotion	Purchase
high	yes	old	own	yes	yes
high	yes	young	rent	yes	no
high	no	old	rent	yes	yes
middle	yes	young	own	yes	yes
high	yes	old	rent	yes	no
high	yes	young	rent	no	no
middle	yes	old	rent	no	no
high	no	young	own	no	yes
high	no	old	own	no	no
high	no	young	rent	no	yes

Uplifting

- Build a classification model *M* using *Purchase* as the class attribute.

Income	Married	Age	Housing	Promotion	Purchase
high	yes	old	own	yes	yes
high	yes	young	rent	yes	no
high	no	old	rent	yes	yes
middle	yes	young	own	yes	yes
high	yes	old	rent	yes	no
high	yes	young	rent	no	no
middle	yes	old	rent	no	no
high	no	young	own	no	yes
high	no	old	own	no	no
high	no	young	rent	no	yes

Uplifting

- In a new consumer dataset,

(1). Set *Promotion* to *yes* for all consumers and predict the purchase probability using ***M***:

Income	Married	Age	Housing	Promotion	Predicted Purchase Prob.
high	yes	young	own	yes	0.75
middle	yes	young	own	yes	0.82
high	no	old	rent	yes	0.70
middle	yes	young	own	yes	0.93
high	no	old	rent	yes	0.78

(2). Set *Promotion* to *no* for all consumers and predict the purchase probability using ***M***:

Income	Married	Age	Housing	Promotion	Predicted Purchase Prob.
high	yes	young	own	no	0.82
middle	yes	young	own	no	0.63
high	no	old	rent	no	0.72
middle	yes	young	own	no	0.81
high	no	old	rent	no	0.75

Uplifting

- For each consumer, calculate the *uplift* as follows:

  uplift = (probability to purchase when promotion is sent)
  - (probability to purchase when promotion is not sent)

Income	Married	Age	Housing	Uplift
high	yes	young	own	- 0.07
middle	yes	young	own	0.19
high	no	old	rent	- 0.02
middle	yes	young	own	0.12
high	no	old	rent	0.03

- Select only those consumers where the uplift is positive and send promotions to only these consumers.

Association Rule Mining

- Typically, two step process
- First, mine all *frequent patterns*
  A frequent pattern is also called a *frequent itemset* or a *large itemset*
- Second, mine *strong rules* from frequent itemsets

Basic Concepts: Frequent Itemsets

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Beer, Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

- itemset: A set of one or more items
- k-itemset: a set of *k* items
- 1-itemset: {beer}, {nuts}, {diaper}, {coffee}, …
- 2-itemset: {beer, nuts}, {beer, coffee}, {eggs, milk}, …
- 3-itemset: {beer, nuts, diaper}, {nuts, coffee, eggs}, …

Basic Concepts: Frequent Itemsets

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Beer, Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

- (*absolute*) *support*, or, *support count* of X: Frequency or the number of transactions that contain itemset X
- (*relative*) *support*, *s*, is the fraction of transactions that contain X (i.e., the probability that a transaction contains X)
- When we say "support" it could mean either. So, interpret it in the context.
- Support of {beer}: 4 (count), 0.8 (4 out of 5), or 80%
- Support of {coffee, diaper}: 2, 0.4 (2 out of 5), or 40%

Basic Concepts: Frequent Itemsets

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Beer, Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

- An itemset X is *frequent* if X's support is no less than a predefined minimum support threshold, *minsup*.
- If *minsup* = 0.6 or 60%
  - {beer} is frequent, or is a frequent itemset, or is a large itemset, or is a frequent pattern
  - {coffee, diaper} is not frequent, or is not a frequent itemset/pattern

Basic Concepts: Association Rules

- A rule R1 = X $\rightarrow$ Y

- Support of R1:

  s(R1) = *support*(X $\cup$ Y),

  or probability that a transaction contains X $\cup$ Y

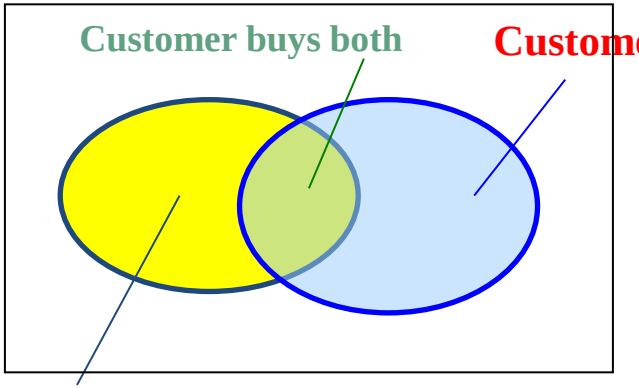
- Confidence of R1:

  c(R1) = *support*(X $\cup$ Y) / *support*(X),

  or conditional probability that a transaction having X also contains Y

Basic Concepts: Association Rules

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Beer, Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



- R1 = {diaper} → {beer}

  s(R1) = support({diaper, beer}) = 3 (count), 0.6 or 60%

  3 transactions (or 60% of all transactions) contain both diaper and beer.

  c(R1) = support({diaper, beer}) / support({diaper})

  = 3 / 4 = 0.75 or 75%

  Among those who purchased diaper, 75% of them also purchased beer.

Basic Concepts: Association Rules

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Beer, Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

- Find all rules $X \rightarrow Y$ with support ≥ minimum support and confidence ≥ minimum confidence. They are called *strong* rules.

- Let *minsup* = 40%, *minconf* = 60%
- Then,
  Beer → *Eggs* (support = 40%, confidence = 50%), is not a strong rule
  Eggs → *Beer* (support = 40%, confidence = 67%), is a strong rule

Apriori Property

- Apriori property of frequent itemsets
  - Any nonempty subset of a frequent itemset must be frequent
  - If {beer, diaper, nuts} is frequent, so is {beer, diaper}.
  - Because every transaction having {beer, diaper, nuts} also contains {beer, diaper}
- It can be also stated as:
  - If an itemset contains a subset that is not frequent, then the itemset can never be frequent.
  - Consider an itemset X = {milk, cheese, egg}.
  - If {milk, egg} is not frequent, then X can never be frequent.

Scalable Mining Methods

- Scalable mining methods: Three major approaches
  - Apriori (Agrawal & Srikant@VLDB'94)
  - Freq. pattern growth (FPgrowth—Han, Pei & Yin @SIGMOD'00)
  - Vertical data format approach (Charm—Zaki & Hsiao @SDM'02)

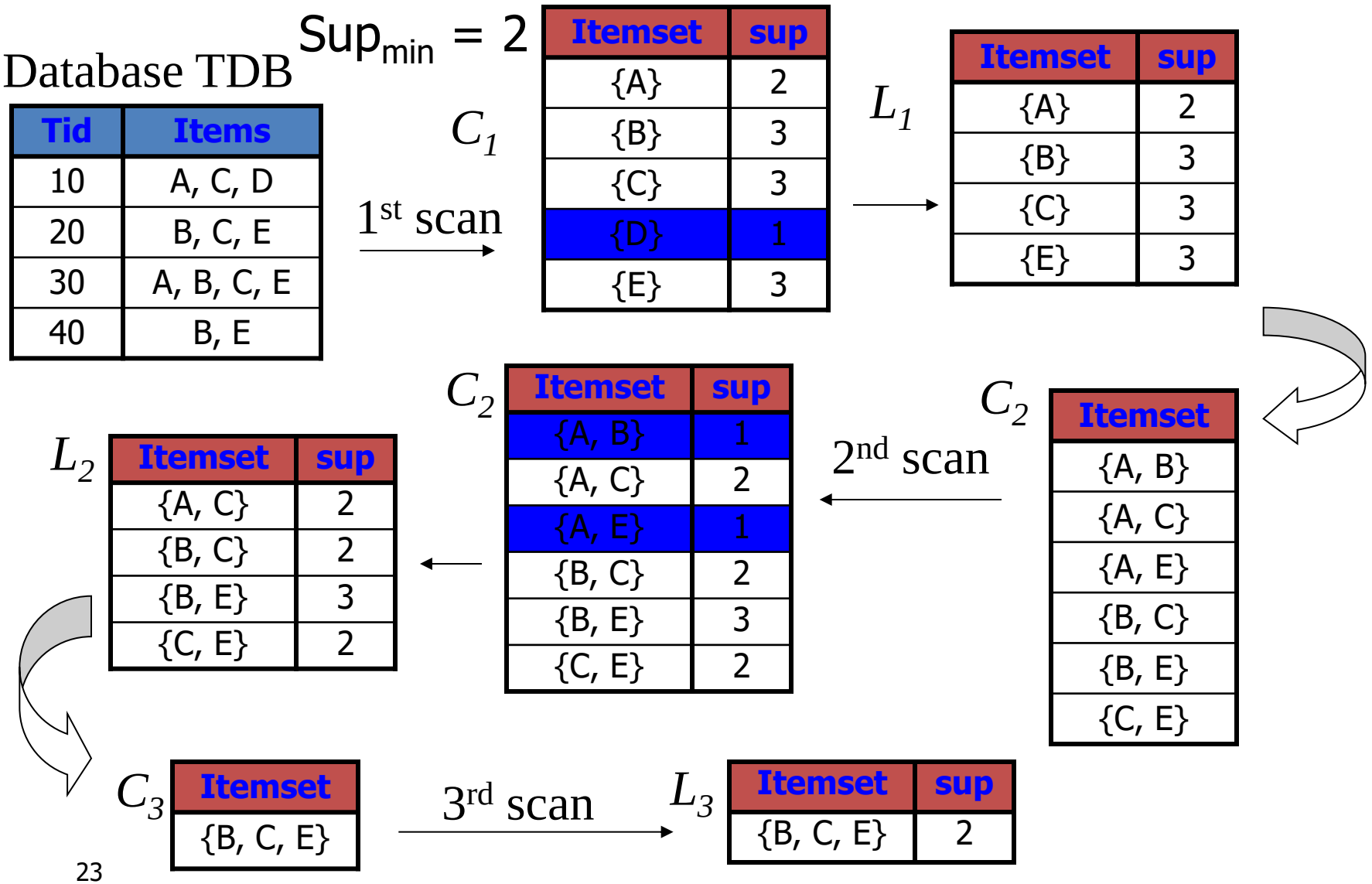
Apriori: A Candidate Generation & Test Approach

- Pruning using Apriori property: If an itemset has a subset which is infrequent, then the itemset should not be tested.

  ("test" means: to determine whether an itemset is frequent or not)

- Algorithm (simplified):
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Prune candidate itemsets using Apriori property
  - Test the candidates against DB
  - Terminate when no frequent or candidate set can be generated

Apriori Algorithm - An Example



Apriori Algorithm (outline)

1. Scan DB and find candidate 1-itemsets: C_1
2. Mine frequent 1-itemsets from $C_1 \rightarrow L_1$
3. Generate candidate 2-itemsets from $L_1 \rightarrow C_2$ (w/o count)
4. Scan DB and count $\rightarrow C_2$ (with count)
5. Mine frequent 2-itemsets from $C_2 \rightarrow L_2$
6. k = 3
7. Generate C_k from L_{k-1} (w/o count)
8. Prune
9. Scan DB and count $\rightarrow C_k$ (with count)
10. Mine L_k from C_k
11. k = k + 1, and Go To Step 7

***. Stop when C_k is empty or L_k is empty

Candidate Itemset Generation

- How to generate C_{k+1} from L_k
  - Join two k-itemsets to generate (k+1)-itmesets
  - When joining two k-itemsets, we join only if the first k-1 items are identical.
- Example: Generate C_4 from L_3.
  - L_3={*abc, abd, acd, ace, bcd*}
    - Generate *abcd* from *abc* and *abd*
    - Generate *acde* from *acd* and *ace*
  - C_4 = {*abcd*, *acde*}

When joining two 3-itemsets, we join only if the first 2 items are identical.

Another Example

Dataset (min. support = 30% or three transactions)

Customer	Items
C1	beer, bread, chip, egg
C2	beer, bread, chip, egg, popcorn,
C3	bread, chip, egg
C4	beer, bread, chip, egg, milk, popcorn
C5	beer, bread, milk
C6	beer, bread, egg
C7	bread, chip, milk
C8	bread, butter, chip, egg, milk
C9	butter, chip, egg

- Items are sorted.

Candidate 1-itemsets

C_1

Itemset	Support count
{beer}	5
{bread}	8
{butter}	2
{chip}	7
{egg}	7
{milk}	4
{popcorn}	2

Frequent 1-itemsets

L_1

Itemset	Support count
{beer}	5
{bread}	8
{chip}	7
{egg}	7
{milk}	4

Candidate 2-itemsets

C_2

Itemset
{beer, bread}
{beer, chip}
{beer, egg}
{beer, milk}
{bread, chip}
{bread, egg}
{bread, milk}
{chip, egg}
{chip, milk}
{egg, milk}

Candidate 2-itemsets with Counts

C_2 (scan the database and count the supports)

Itemset	Support count
{beer, bread}	5
{beer, chip}	3
{beer, egg}	4
{beer, milk}	2
{bread, chip}	6
{bread, egg}	6
{bread, milk}	4
{chip, egg}	6
{chip, milk}	3
{egg, milk}	2

Frequent 2-itemsets

L_2

Itemset	Support count
{beer, bread}	5
{beer, chip}	3
{beer, egg}	4
{bread, chip}	6
{bread, egg}	6
{bread, milk}	4
{chip, egg}	6
{chip, milk}	3

Candidate 3-itemsets

C_3

Itemset
{beer, bread, chip}
{beer. bread, egg}
{beer, chip, egg}
{bread, chip, egg}
{bread, chip, milk}
{bread, egg, milk}
{chip, egg, milk}

Two frequent 2-itemsets are joined only if the first items are identical.

We join {beer, bread} and {beer, chip} to generate {beer, bread, chip}.

But, we do not join {beer, chip} and {chip, egg}.

Candidate 3-itemsets

C_3

Itemset
{beer, bread, chip}
{beer. bread, egg}
{beer, chip, egg}
{bread, chip, egg}
{bread, chip, milk}
{bread, egg, milk}
{chip, egg, milk}

C_3 after pruning

Itemset
{beer, bread, chip}
{beer. bread, egg}
{beer, chip, egg}
{bread, chip, egg}
{bread, chip, milk}

{egg, milk} is not frequent (i.e., not in L2).
{bread, egg, milk} and {chip, egg, milk} are pruned

Candidate 3-itemsets with Supports

C_3 (scan the database and count supports)

Itemset	Support count
{beer, bread, chip}	3
{beer. bread, egg}	4
{beer, chip, egg}	3
{bread, chip, egg}	5
{bread, chip, milk}	3

Frequent 3-itemsets

L_3

Itemset	Support count
{beer, bread, chip}	3
{beer. bread, egg}	4
{beer, chip, egg}	3
{bread, chip, egg}	5
{bread, chip, milk}	3

Candidate 4-itemsets

C_4

Itemsets
{beer, bread, chip, egg}
{bread, chip, egg, milk}

C_4 after pruning

Itemsets
{beer, bread, chip, egg}

Again, we join two frequent 3-itemsets only if the first *two* items are identical.

{beer, bread, chip} JOIN {beer, bread, egg} generates {beer, bread, chip, egg}
{bread, chip, egg} JOIN {bread, chip, milk} generates {bread, chip, egg, milk}

Prune: {chip, egg, milk} is not frequent. So, {bread, chip, egg, milk} is pruned.

Candidate 4-itemsets

C_4 (scan the database and count the supports)

Itemsets	Support count
{beer, bread, chip, egg}	3

Frequent 4-itemsets

L_4

Itemsets	Support count
{beer, bread, chip, egg}	3

C_5 is empty. Stop.

All Frequent Itemsets

Frequent itemsets $L = L_1 \cup L_2 \cup L_3 \cup L_4$

L = {{beer}, {bread}, {chip}, {egg}, {milk}, {beer, bread},
{beer, chip}, {beer, egg}, {bread, chip}, {bread, egg},
{bread, milk}, {chip, egg}, {chip, milk}, {beer, bread, chip},
{beer. bread, egg}, {beer, chip, egg}, {bread, chip, egg},
{bread, chip, milk}, {beer, bread, chip, egg}}

Mining Strong Rules

- For each frequent itemset, identify all nonempty proper subsets:
- Example: from {beer, bread, egg}
- All nonempty proper subsets are:

  {beer}, {bread}, {egg}, {beer, bread}, {beer, egg}, {bread, egg}
- For each subset, we form a rule:

  R1: {beer} $\Rightarrow$ {bread, egg}

  R2: {bread} $\Rightarrow$ {beer, egg}

  R3: {egg} $\Rightarrow$ {beer, bread}

  R4: {beer, bread} $\Rightarrow$ {egg}

  R5: {beer, egg} $\Rightarrow$ {bread}

  R6: {bread, egg} $\Rightarrow$ {beer}

Mining Strong Rules

- Compute the confidences:

confidence = sup(all items) / sup(antecedent)

conf(R1) = (sup({ beer, bread, egg})) / sup({beer}) = 4/5 = 80%

conf(R2) = (sup({ beer, bread, egg})) / sup({bread})= 4/8 = 50%

conf(R3) = (sup({ beer, bread, egg})) / sup({egg}) = 4/7 = 57.1%

conf(R4) = (sup({ beer, bread, egg})) / sup({beer, bread}) = 4/5 = 80%

conf(R5) = (sup({ beer, bread, egg})) / sup({beer, egg}) = 4/4 = 100%

conf(R6) = (sup({ beer, bread, egg})) / sup({bread, egg}) = 4/6 = 66.7%

Mining Strong Rules

- Choose the rules whose confidences satisfy minimum confidence.

- If *min_conf* = 80%, R1, R4, and R5 are strong rules.
- If *min_conf* = 60%, R1, R4, R5, and R6 are strong rules.

Exercise

- Mine all frequent itemsets from the following dataset. Assume that the minimum support is 30% (or 3 transactions).
- Then, mine all strong rules from the first frequent 3-itemset (when 3-itemsets are sorted by the items). Assume that the minimum confidence is 80%.

TID	Items
100	2,4,5,6
200	1,4,5,7
300	2,4,5
400	1,2,4,5,6,7
500	1,2,6
600	1,2,5,7
700	2,4,6
800	2,3,4,5,6
900	3,4,5,6

Collaborative Filtering

- Generate recommendation to a user using the following information, in general:
  - What the user purchased in the past.
  - Which items they have in the shopping cart.
  - Which items they rated high or liked.
  - What other users have purchased

- Will discuss user-based collaborative filtering (UBCF) and item-based collaborative filtering (IBCF).

Collaborative Filtering

- Data format
- Assume: *n* users $U = (U_1, U_2, \ldots, U_n)$, *P* items $I = (I_1, I_2, \ldots, I_p)$

	Item ID			
User ID	I_1	I_2	$\cdots$	I_p
U_1	$r_{1,1}$	$r_{1,2}$	$\cdots$	$r_{1,p}$
U_2	$r_{2,1}$	$r_{2,2}$	$\cdots$	$r_{2,p}$
$\vdots$				
U_n	$r_{n,1}$	$r_{n,2}$	$\cdots$	$r_{n,p}$

$r_{i,j}$ is the rating of Item I_j by user U_i

Collaborative Filtering

- User-based collaborative filtering
  - Predict the rating of an item by a user *U* based on other users that are similar to *U*.

- Item-based collaborative filtering
  - Predict the ratings of an item *I* based on other items that are similar to *I.*

- Similar users or similar items are identified using a similarity measure.
- Two typical similarity measures: Pearson's correlation and cosine similarity
- Will Illustrate with a small example dataset using Pearson's correlation.

Collaborative Filtering

- UBCF:
  - Calculate correlation between U1 and all other users (use only items that are co-rated).
  - Predcition:
    - r(U1, I1) is predicted as weighted sum of other user's prediction of I1.
    - Correlations are used as weights.
    - Adjusted for user bias.
    - Normalized.

Collaborative Filtering

- Prediction:

  Prediction of user *a*'s rating of item *j*:

  $$P_{a,j} = \overline{r_a} + \frac{\sum_i ((r_{i,j} - \overline{r_i}) * (corr(a,i))}{\sum_i |corr(a,i)|}$$

  - *i*: all other users (except user *a*)
  - *corr*(*a*, *i*): correlation between user *a* and user *i*
  - $\overline{r_a}$: average rating of user *a*
  - $r_{i,j}$: rating of item *j* by user *i*
  - $\overline{r_i}$: average rating of user *i*

Collaborative Filtering

- Example dataset:

	I1	I2	I3	I4
U1	?	3	2	4
U2	3	2	5	2
U3	5	5	4	2
U4	5	3	2	4

Collaborative Filtering

- Example:

  corr(U1,U2) = -0.866

  corr(U1,U3) = -0.655

  corr(U1,U4) = 1

	I1	I2	I3	I4	Avg
U1	?	3	2	4	3
U2	3	2	5	2	3
U3	5	5	4	2	4
U4	5	3	2	4	3.5

  Prediction

  P(1,1) = $\overline{r_1}$ + (x / y)

  $\overline{r_1}$ = 3

  x = (3-3)*(-0.866) + (5-4)*(-0.655) + (5-3.5)*1 = 0.845

  y = |-0.866| + |-0.655| + |1| = 2.51

  P(1,1) = 3 + (0.845 / 2.51) = 3.335

Collaborative Filtering

- IBCF:
  - Calculate correlation between item *I*1 and all other items (use only items that are co-rated).
  - Prediction:
    - r(U1, I1) is predicted as weighted sum of ratings of all items by user U1.
    - Correlations are used as weights.
    - Normalized.

Collaborative Filtering

- Prediction:

  Prediction of user *a*'s rating of item *i*:

$$P_{a,i} = \overline{r_i} + \frac{\sum_j (r_{a,j} - \overline{r_j}) * corr(i,j)}{\sum_j |corr(i,j)|}$$

  - *j*: all other items (except item *i*)
  - *corr*(*i*, *j*): correlation between item *i* and item *j*
  - $\overline{r_i}$: average rating on item *i*
  - $r_{a,j}$: rating of item *j* by user *a*
  - $\overline{r_j}$: average rating on item *j*

Collaborative Filtering

- Example:

  corr(I1,I2) = 0.756

  corr(I1,I3) = -0.756

  corr(I1,I4) = 0.5

	I1	I2	I3	I4
U1	?	3	2	4
U2	3	2	5	2
U3	5	5	4	2
U4	5	3	2	4
Avg	4.33	3.25	3.25	3

  Prediction

  P(1,1) = x / y

  x = (3 – 3.25) * 0.756 + (2 – 3.25) * (-0.756) + (4 – 3) * 0.5

  = 1.256

  y = |0.756| + |-0.756| + |0.5| = 2.012

  P(1,1) = 4.33 + (1.256 / 2.012) = 4.954

Collaborative Filtering

- Usually we use only top-k users or top-k items (based on similarities).
- Recommendation for a user *a*:
  - For each item which the user *a* did not rate, we calculate the prediction.
  - Select and recommend top-N items based on the predicted ratings.

Association Rules vs. Collaborative Filtering

- AR: focus entirely on frequent (popular) **item combinations**. Data rows are single transactions. Ignores user dimension. Often used in displays (what goes with what).
- CF: focus is on **user preferences**. Data rows are user purchases or ratings over time. Can capture "long tail" of user preferences – useful for recommendations involving unusual items

References

- Galit Shmueli et al., "Machine Learning for Business Analytics: Concepts, Techniques, and Applications in R," Second Ed. 2023, Wiley