

Week 9-10: Hybrid Recommender Systems and Evaluation Metrics

CSX4207/ITX4207: Decision Support
and Recommender Systems

Asst. Prof. Dr. Rachsuda Setthawong

Objectives

- To understand concept of hybridizing recommender systems
- To understand different types of Hybrid Recommender Systems
- To be able to create simple Hybrid Recommender Systems
- To understand evaluation metrics

Outlines

- Hybrid Recommender Systems
- Types of Hybrid Recommender Systems
 - Monolithic Hybridization Design
 - Parallelized Hybridization Design
 - Pipelined Hybridization Designs
- Evaluation Metrics

Hybrid Recommender Systems

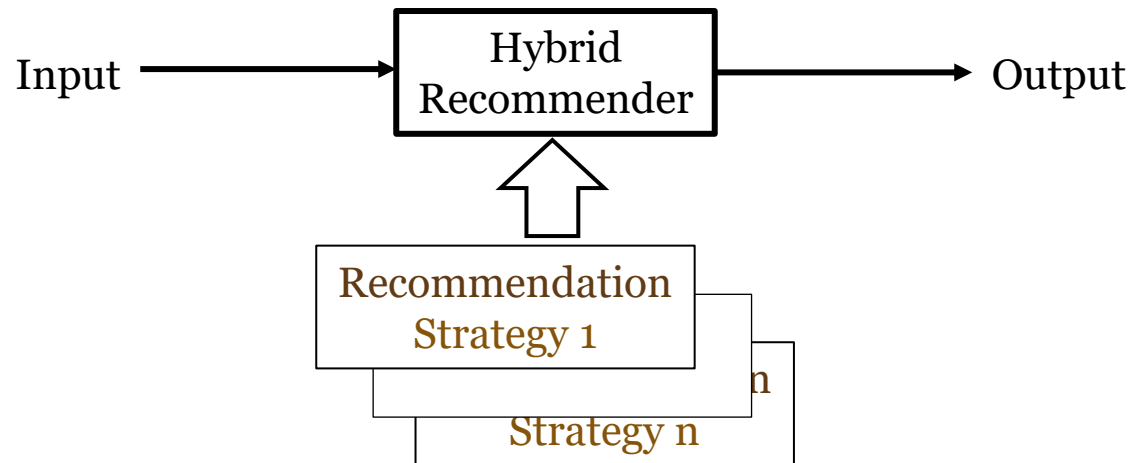
- *hybrida* [lat.]: an object made by combining two different elements
- Main Idea:
 - Cross two (or more) species/implementations.
 - Combine the strengths of the three base techniques to overcome their weak points and problems.
 - Achieve desirable properties not present in parent individuals

Types of Hybrid Recommender Systems

- **Monolithic Hybridization Design**
 - **Feature Combination**
 - **Feature Augmentation**
- **Parallelized Hybridization Design**
- **Pipelined Hybridization Designs**

Monolithic Hybridization Design

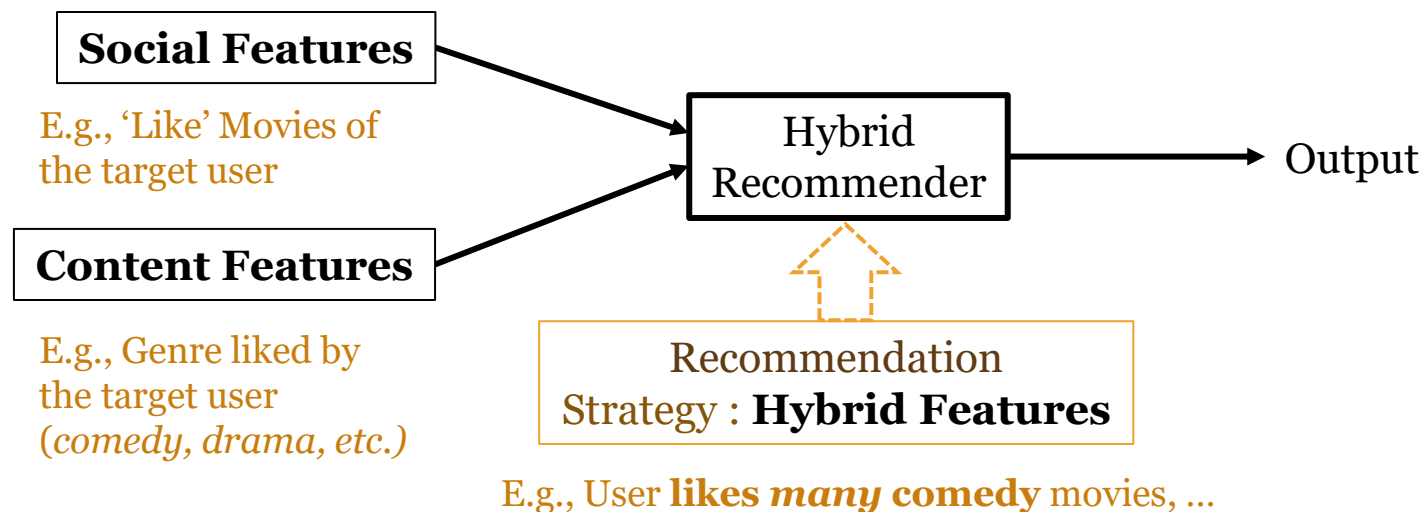
- Only a single recommendation component



- “**Virtual**” hybridization
 - Combine *Features/knowledge sources* of different paradigms.

Monolithic Hybridization Designs: Feature Combination

- Combine knowledge from several sources, *e.g., ratings, user demographics* and/or explicit requirements in computing similarity.
- "Hybrid" content features:



Example 1 of Feature Combination (1)

- Rule for deriving hybrid features:
 - IF a user bought **mainly** items of genre X, THEN set the user characteristic ***'User likes many X items'*** to true.
- Input data:

Genre		romance	mystery	mystery	mystery	fiction
User	Item1	Item2	Item3	Item4	Item5	
Alice		1		1		
User1		1	1			1
User2	1	1				1
User3	1		1			
User4						1

Item	Genre
Item1	romance
Item2	mystery
Item3	mystery
Item4	mystery
Item5	fiction

Example 1 of Feature Combination (2)

Genre *romance* *mystery* *mystery* *mystery* *fiction*

User	Item1	Item2	Item3	Item4	Item5
Alice		1		1	
User1		1	1		1
User2	1	1			1
User3	1		1		
User4					1

Rule for deriving hybrid features:

- IF # of purchase ≥ 2
THEN 'many' = true
- IF # of purchase = 1
THEN 'some' = true

User	Alice	User1	User2	User3	User4
User likes many mystery books	true	true			
User likes some mystery books			true	true	
User likes many romance books					
User likes some romance books			true	true	
User likes many fiction books					
User likes some fiction books		true	true		true

hybrid features

Example 1 of Feature Combination (3)

- Who is the most similar user to Alice?

Genre	romance	mystery	mystery	mystery	fiction
User	Item1	Item2	Item3	Item4	Item5
Alice		1		1	
User1		1	1		1
User2	1	1			1
User3	1		1		
User4					1

Example 1 of Feature Combination (4)

- Who is the most similar user to Alice wrt the combined features?

Genre romance **mystery** mystery **mystery** fiction

User	Item1	Item2	Item3	Item4	Item5
Alice		1		1	
User1		1	1		1
User2	1	1			1
User3	1		1		
User4					1

Rule for deriving hybrid features:

- IF # of purchase ≥ 2
THEN 'many' = true
- IF # of purchase = 1
THEN 'some' = true

User	Alice	User1	User2	User3	User4
User likes many mystery books	true	true			
User likes some mystery books			true	true	
User likes many romance books					
User likes some romance books			true	true	
User likes many fiction books					
User likes some fiction books		true	true		true

Example 2 of Feature Combination (1)

- Who is the most similar user to Alice *NOT considering* priority of the features?

User	Navigation Action (f1)	Click on item's detail pages (f2)	Contextual user requirements (f3)	Actual purchase (f4)
Alice	n3, n4	i5	k5	∅
User1	n1, n5	i3, i5	k5	i1
User2	n3, n4	i3, i5, i7	∅	i3
User3	n2, n3, n4	i2, i4, i5	k2, k4	i4

Example 2 of Feature Combination (2)

- Who is the most similar user to Alice *NOT* considering **priority** of the features?

User	Navigation Action (f1)	Click on item's detail pages (f2)	Contextual user requirements (f3)	Actual purchase (f4)
Alice	n3, n4	i5	k5	∅
User1	n1, n5	i3, i5	k5	i1
User2	n3, n4	i3, i5, i7	∅	i3
User3	n2, n3, n4	i2, i4, i5	k2, k4	i4

Example 2 of Feature Combination (3)

- Suppose that the priority of the features are as follows: $(f_3, f_4) > f_2 > f_1$,
 - who is the most similar user to Alice *considering* the **priority** of the features?

User	Navigation Action (f₁)	Click on item's detail pages (f₂)	Contextual user requirements (f₃)	Actual purchase (f₄)
Alice	n3, n4	i5	k5	∅
User1	n1, n5	i3, i5	k5	i1
User2	n3, n4	i3, i5, i7	∅	i3
User3	n2, n3, n4	i2, i4, i5	k2, k4	i4

Example 2 of Feature Combination (4)

- Suppose that the priority of the features are as follows: $(f_3, f_4) > f_2 > f_1$,
 - who is the most similar user to Alice *considering* the **priority** of the features?

User	Navigation Action (f₁)	Click on item's detail pages (f₂)	Contextual user requirements (f₃)	Actual purchase (f₄)
Alice	n3, n4	i5	k5	∅
User1	n1, n5	i3, i5	k5	i1
User2	n3, n4	i3, i5, i7	∅	i3
User3	n2, n3, n4	i2, i4, i5	k2, k4	i4

Monolithic Hybridization Designs:

Feature Augmentation

- Main Idea: apply *more complex transformation steps* to combine several types of inputs.
 - *Augments the **feature space** of the actual recommender by preprocessing its knowledge sources.*

An Example of Feature Augmentation: Content-boosted collaborative filtering (1)

- Predicts a **target user's rating** based on a **collaborative mechanism** *that includes content-based predictions*.
 - What is Alice's rating on item5?

User	User rating on item5 ($v_{User,Item5}$)	Pearson Coeff. ($P_{Alice,User}$)	No. of user ratings (n_{User})	No. of overlapping rating ($n_{Alice,User_i}$)
Alice	?		40	
User1	4	0.8	14	6
User2	2.2	0.7	55	28

An Example of Feature Augmentation: Content-boosted collaborative filtering (2)

- Step 1: create a pseudo-user-ratings vector $\mathbf{v}_{u,i}$:

$$v_{u,i} = \begin{cases} r_{u,i} & \text{: if user } u \text{ rated item } i \\ c_{u,i} & \text{else content-based prediction} \end{cases}$$

User	User rating on item5 ($\mathbf{v}_{\text{User}, \text{Item5}}$)	Pearson Coeff. ($P_{\text{Alice}, \text{User}}$)	No. of user ratings (n_{User})	No. of overlapping rating ($n_{\text{Alice}, \text{User}_i}$)
Alice	?		40	
User1	4	0.8	14	6
User2	2.2	0.7	55	28

Rating that user1
gives to item5.

Obtained from content-
based prediction for
User2 on item5.

An Example of Feature Augmentation:

Content-boosted collaborative filtering (3)

- Step 2: **compute predictions** based on the pseudo ratings:

Self-weighting on
The content-based prediction
on item5

Adjust user's rating based on
Correlation between the target user
and the most similar user(s)

$$rec_{cbcf}(a, i) = \frac{sw_a c_{a,i} + \sum_{\substack{u=1 \\ u \neq a}}^n hw_{a,u} P_{a,u} v_{u,i}}{sw_a + \sum_{\substack{u=1 \\ u \neq a}}^n hw_{a,u} P_{a,u}}$$

Target user

item

User's rating on item5
($v_{User, Item5}$)

Pearson Coeff.
($P_{Alice, User}$)

An Example of Feature Augmentation: Content-boosted collaborative filtering (4)

Self-weighting on
The content-based prediction
on item5

content-based prediction
on item i

Assume that $C_{\text{Alice}, \text{Item5}} = 3$

Target user: a , item: i

$$rec_{cbcf}(a, i) = \frac{sw_a c_{a,i} + \sum_{u \neq a}^n hw_{a,u} P_{a,u} v_{u,i}}{sw_a + \sum_{u \neq a}^n hw_{a,u} P_{a,u}}$$

User	No. of user ratings (n_{User})
Alice	40
User1	14
User2	55

Self-weighting factor:
(confidence wrt no. of
original rating values of the
target user a)

$$sw_i = \begin{cases} \frac{n_i}{50} \times \max & : \text{if } n_i < 50 \\ \max & : \text{Otherwise} \end{cases}$$

Assume that $\max = 2$ (in the paper)

$$sw_{\text{Alice}} = \frac{n_{\text{Alice}}}{50} \times \max = \frac{40}{50} \times 2 = 1.6$$

An Example of Feature Augmentation: Content-boosted collaborative filtering (5)

Adjust user's rating based on
Correlation between the target user and
the most similar user(s)

Target user

item

$$rec_{cbcf}(a, i) = \frac{sw_a c_{a,i} + \sum_{u \neq a}^n hw_{a,u} P_{a,u} v_{u,i}}{sw_a + \sum_{u \neq a}^n hw_{a,u} P_{a,u}}$$

Pseudo-user-ratings
(from step 1)

Pearson correlation coeff.

Hybrid correlation weight
(calculation shown in the next slide)

User	User rating on item5 ($v_{User, Item5}$)	Pearson Coeff. ($P_{Alice, User}$)
Alice	?	
User1	4	0.8
User2	2.2	0.7

Hybrid correlation weight

User	User rating on item5 ($v_{User,Item5}$)	Pearson Coeff. ($P_{Alice,User}$)	No. of user ratings (n_{User})	No. of overlapping rating ($n_{Alice,User}$)
Alice	?		40	
User1	4	0.8	14	6
User2	2.2	0.7	55	28

$$hw_{a,u} = sg_{a,u} + hm_{a,u}$$

Significant
Weighting
Factor ($sg_{a,u}$)

$$sg_{a,u} = \begin{cases} \frac{n_{a,u}}{50} & : \text{if } n_{a,u} < 50 \\ 1 & : \text{Otherwise} \end{cases}$$

Harmonic
Mean
Weighting
Factor ($hm_{a,u}$)

$$hm_{a,u} = \frac{2m_a m_u}{m_a + m_u}$$

$$m_i = \begin{cases} \frac{n_i}{50} & : \text{if } n_i < 50 \\ 1 & : \text{Otherwise} \end{cases}$$

$$hw_{Alice,User1} = sg_{Alice,User1} + hm_{Alice,User1} = 0.12 + 0.415 = 0.535$$

$$sg_{Alice,User1} = \frac{n_{Alice,User1}}{50} = \frac{6}{50} = 0.12$$

$$hm_{Alice,User1} = \frac{2m_{Alice}m_{User1}}{m_{Alice}+m_{User1}} = \frac{2 \times \frac{40}{50} \times \frac{14}{50}}{\frac{40}{50} + \frac{14}{50}} = \frac{2 \times 0.8 \times 0.28}{0.8 + 0.28} = 0.415$$

An Example of Feature Augmentation (6)

- Step 2: compute predictions based on the pseudo ratings:

The diagram illustrates the formula for $rec_{cbcf}(a, i)$ and its components. The formula is:

$$rec_{cbcf}(a, i) = \frac{sw_a c_{a,i} + \sum_{\substack{u=1 \\ u \neq a}}^n hw_{a,u} P_{a,u} v_{u,i}}{sw_a + \sum_{\substack{u=1 \\ u \neq a}}^n hw_{a,u} P_{a,u}}$$

Annotations for the formula components:

- Target user**: points to a in $rec_{cbcf}(a, i)$
- item**: points to i in $rec_{cbcf}(a, i)$
- Self-weighing factor**: points to sw_a in the numerator and denominator
- content-based prediction on item i** : points to $c_{a,i}$ in the numerator
- Pearson correlation coeff.**: points to $P_{a,u}$ in the numerator and denominator
- Hybrid correlation weight**: points to $hw_{a,u}$ in the numerator and denominator
- Pseudo-user-ratings**: points to $v_{u,i}$ in the numerator

A numerical example is provided below the formula:

$$rec_{cbcf}(Alice, Item5) = \frac{1.6 \times 3 + (0.535 \times 0.8 \times 4 + 1.45 \times 0.7 \times 2.2)}{1.6 + (0.535 \times 0.8 + 1.45 \times 0.7)} = 2.87$$

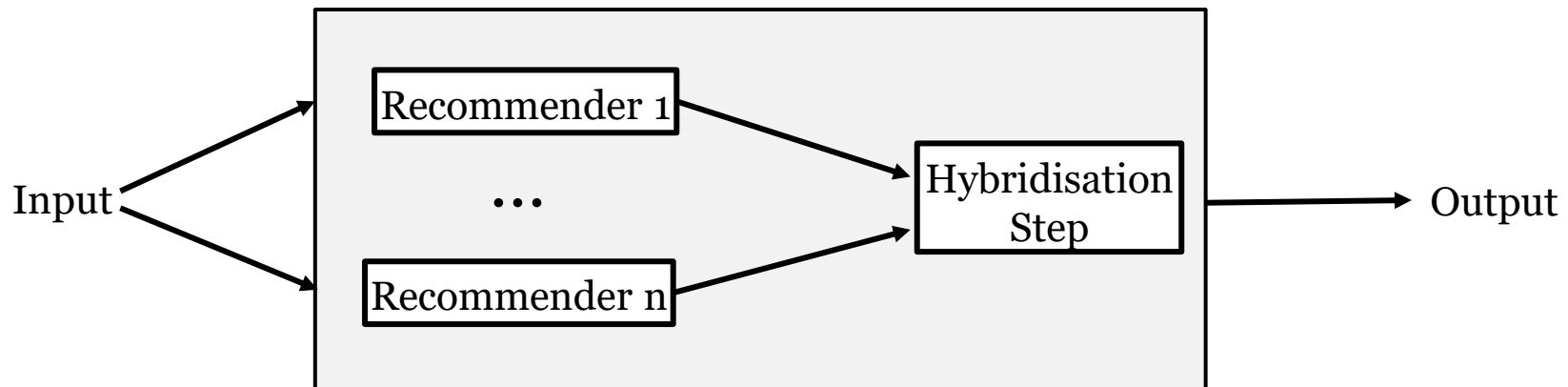
The example uses the value $c_{Alice, Item5}$ for the content-based prediction.

Types of Hybrid Recommender Systems

- Monolithic Hybridization Design
- Parallelized Hybridization Design
 - Weighted
 - Switching
- Pipelined Hybridization Designs

Parallelized Hybridization Design

- **Run** several recommendation algorithms (*in **parallel***) and **combine** results to generate final recommendations.
- May apply **weighting (voting)** for result combination.
 - **Approach 1:** feasibly **learn weights dynamically**.
 - **Approach 2:** use **switching** if weights of only one algorithm is one; the rest algorithm(s) has zero weight.



Parallelized Hybridization Design: Computing Weighted Recommendations

$$rec_{weighted}(u, i) = \sum_{k=1}^n \beta_k \times rec_k(u, i)$$

No. of recommenders n

Weight β_k

Weighted recommendation on item i for the active user u

Recommender 1		
Item1	0.5	1
Item2	0	
Item3	0.3	2
Item4	0.1	3
Item5	0	

Recommender 2		
Item1	0.8	2
Item2	0.9	1
Item3	0.4	3
Item4	0	
Item5	0	

$$[(0.5 \times 0.5) + (0.5 \times 0.8)]$$

Recommender weighted (0.5 : 0.5)		
Item1	0.65	1
Item2	0.45	2
Item3	0.35	3
Item4	0.05	4
Item5	0.00	

Note:

1. Item scores must be in the same range.
2. $\sum \beta_k = 1$

Parallelized Hybridization Design:

Weighting Strategies

- Empirical bootstrapping
 - Require historic data.
 - **Generate recommendations** using **different weightings** and **select the one** with the **best** result.
- Dynamic adjustment of weights
 - Initially use *uniform weight distribution*.
 - Iteratively *adjust weights to minimize prediction error*.

Parallelized Hybridization Design:

Caution when Applying **Weighted (1)**

- Suppose that Alice bought items 1 and 4 (= *actual rating* = ($r_i = 1$))
 - Select weighting with smallest Mean Absolute Error (MAE)

$$MAE = \frac{\sum_{r_i \in R} \sum_{k=1}^n \beta_k \times |rec_k(u, i) - r_i|}{|R|}$$

Difference between actual and predicted ratings

Absolute Errors and MAE

Beta1	Beta2		Predicted rating on Rec1	Predicted rating on Rec2	error	MAE
0.1	0.9	Item1	0.5	0.8	0.23	0.61
		Item4	0.1	0.0	0.99	
0.3	0.7	Item1	0.5	0.8	0.29	0.63
		Item4	0.1	0.0	0.97	
0.5	0.5	Item1	0.5	0.8	0.35	0.65
		Item4	0.1	0.0	0.95	
0.7	0.3	Item1	0.5	0.8	0.41	0.67
		Item4	0.1	0.0	0.93	
0.9	0.1	Item1	0.5	0.8	0.47	0.69
		Item4	0.1	0.0	0.91	

Selected weights with best MAE (lowest)

low

high

Parallelized Hybridization Design:

Caution when Applying **Weighted (2)**

- Suppose that Alice bought items 1 and 4 (= *actual rating* = ($r_i = 1$))
 - Select weighting with smallest Mean Absolute Error (MAE)

Absolute difference between
actual and predicted ratings

$$MAE = \frac{\sum_{r_i \in R} \sum_{k=1}^n \beta_k \times |rec_k(u, i) - r_i|}{|R|}$$

$$\text{Error_item1}[\text{beta}(0.1, 0.9)] = (0.1 \times |0.5 - 1|) + (0.9 \times |0.8 - 1|) = 0.23$$

Absolute Errors and MAE						
Beta1	Beta2		Predicted rating on Rec1	Predicted rating on Rec2	error	MAE
0.1	0.9	Item1	0.5	0.8	0.23	0.61
		Item4	0.1	0.0	0.99	

Parallelized Hybridization Design:

Caution when Applying **Weighted (3)**

- Suppose that Alice bought items 1 and 4 (= *actual rating* = ($r_i = 1$))
 - Select weighting with smallest Mean Absolute Error (MAE)

Absolute difference between
actual and predicted ratings

$$MAE = \frac{\sum_{r_i \in R} \sum_{k=1}^n \beta_k \times |rec_k(u, i) - r_i|}{|R|}$$

$$\begin{aligned} MAE(\text{beta}(0.1, 0.9)) &= \text{error_item1} + \text{error_item4} = \\ &= \{ [(0.1 \times |0.5 - 1|) + (0.9 \times |0.8 - 1|)] + [(0.1 \times |0.1 - 1|) + (0.9 \times |0.0 - 1|)] \} / 2 \\ &= (0.23 + 0.99) / 2 = 0.61 \end{aligned}$$

Absolute Errors and MAE						
Beta1	Beta2		Predicted rating on Rec1	Predicted rating on Rec2	error	MAE
0.1	0.9	Item1	0.5	0.8	0.23	0.61
		Item4	0.1	0.0	0.99	

Parallelized Hybridization Design: **Switching**

- $\exists_1 k : 1 \dots n \text{ } rec_{switching}(u, i) = rec_k(u, i)$
- Equivalent to dynamic weights when one β equals 1 and the rest β s equal 0.
- An example:

IF too few ratings in the system THEN

use knowledge-based

ELSE

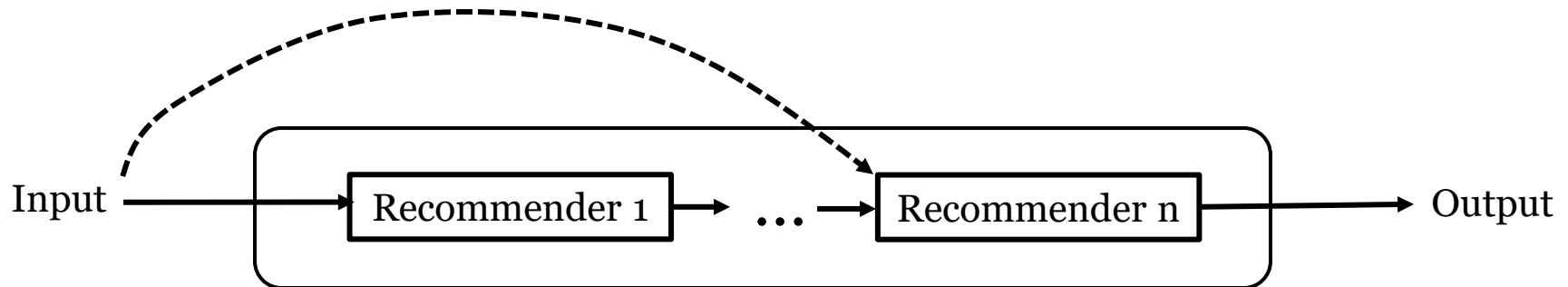
use collaborative-based

Types of Hybrid Recommender Systems

- Monolithic Hybridization Design
- Parallelized Hybridization Design
- **Pipelined Hybridization Designs**
 - Cascade

Pipelined Hybridization Designs

- Sequentially run recommender systems such that a successor alters results from its predecessor.
- Two designs
 - Cascade
 - Meta-level



Pipelined Hybridization Designs: Cascade (1)

- Generating successor's recommendations:

$$rec_{cascade}(u, i) = rec_n(u, i)$$

where $\forall k \geq 2$

$$rec_k(u, i) = \begin{cases} rec_k(u, i) & : rec_{k-1}(u, i) \neq 0 \\ 0 & : \text{else} \end{cases}$$

- Subsequent** recommender **either alters** score of its predecessor's items **or discards** them.
 - An item will be **suggested** by the k^{th} technique **only if** the $(k-1)^{\text{th}}$ technique **also** assigned a **nonzero score** to it.

Pipelined Hybridization Designs: Cascade (2)

Recommender 1			Recommender 2		
Item1	0.5	1	Item1	0.8	2
Item2	0		Item2	0.9	1
Item3	0.3	2	Item3	0.4	3
Item4	0.1	3	Item4	0	
Item5	0		Item5	0	

Recommender cascaded (rec1, rec2)	
Item1	0.5
Item2	0
Item3	0.3
Item4	0.1
Item5	0

- Examples of applying cascade RS:
 - Recommender 1 discards some items.
 - Recommender 2 re-calculating their score.

Pipelined Hybridization Designs: Cascade (3)

Recommender 1			Recommender 2		
Item1	0.5	1	Item1	0.8	2
Item2	0		Item2	0.9	1
Item3	0.3	2	Item3	0.4	3
Item4	0.1	3	Item4	0	
Item5	0		Item5	0	

Recommender cascaded (rec1, rec2)	
Item1	0.8
Item2	0
Item3	0.3
Item4	0.1
Item5	0

- Examples of applying cascade RS:
 - Recommender 1 discards some items.
 - Recommender 2 re-calculating their score.

Pipelined Hybridization Designs: Cascade (4)

Recommender 1			Recommender 2		
Item1	0.5	1	Item1	0.8	2
Item2	0		Item2	0.9	1
Item3	0.3	2	Item3	0.4	3
Item4	0.1	3	Item4	0	
Item5	0		Item5	0	

Recommender cascaded (rec1, rec2)	
Item1	0.8
Item2	0
Item3	0.3
Item4	0.1
Item5	0

No change

- Examples of applying cascade RS:
 - Recommender 1 discards some items.
 - Recommender 2 re-calculating their score.

Pipelined Hybridization Designs: Cascade (5)

Recommender 1			Recommender 2		
Item1	0.5	1	Item1	0.8	2
Item2	0		Item2	0.9	1
Item3	0.3	2	Item3	0.4	3
Item4	0.1	3	Item4	0	
Item5	0		Item5	0	

Recommender cascaded (rec1, rec2)	
Item1	0.8
Item2	0
Item3	0.4
Item4	0.1
Item5	0

- Examples of applying cascade RS:
 - Recommender 1 discards some items.
 - Recommender 2 re-calculating their score.

Pipelined Hybridization Designs: Cascade (6)

Recommender 1			Recommender 2		
Item1	0.5	1	Item1	0.8	2
Item2	0		Item2	0.9	1
Item3	0.3	2	Item3	0.4	3
Item4	0.1	3	Item4	0	
Item5	0		Item5	0	

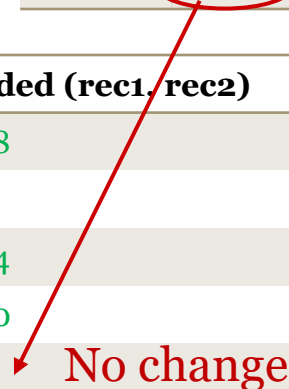
Recommender cascaded (rec1, rec2)	
Item1	0.8
Item2	0
Item3	0.4
Item4	0.0
Item5	0

- Examples of applying cascade RS:
 - Recommender 1 discards some items.
 - Recommender 2 re-calculating their score.

Pipelined Hybridization Designs: Cascade (7)

Recommender 1			Recommender 2		
Item1	0.5	1	Item1	0.8	2
Item2	0		Item2	0.9	1
Item3	0.3	2	Item3	0.4	3
Item4	0.1	3	Item4	0	
Item5	0		Item5	0	

Recommender cascaded (rec1, rec2)		
Item1		0.8
Item2		0
Item3		0.4
Item4		0.0
Item5		0

 No change

- Examples of applying cascade RS:
 - Recommender 1 discards some items.
 - Recommender 2 re-calculating their score.

Pipelined Hybridization Designs: Cascade (8)

Recommender 1		
Item1	0.5	1
Item2	0	
Item3	0.3	2
Item4	0.1	3
Item5	0	

Recommender 2		
Item1	0.8	2
Item2	0.9	1
Item3	0.4	3
Item4	0	
Item5	0	

- Examples of applying cascade RS:
 - Recommender 1 discards some items.
 - Recommender 2 re-calculating their score.

Recommender cascaded (rec1, rec2)		
Item1	0.8	1
Item2	0	
Item3	0.4	2
Item4	0.0	
Item5	0	

New ranks

Outlines

- Hybrid Recommender Systems
- Types of Hybrid Recommender Systems
 - Monolithic Hybridization Design
 - Parallelized Hybridization Design
 - Pipelined Hybridization Designs
- Evaluation Metrics

Methodology

- **N-fold cross-validation:**
 - A stratified random selection technique
 - **(N-1) out of N folds** of user profiles is used *for model building*.
 - **1 out of N folds** of user profiles is used *for evaluation*.
 - **Repeat** them for **N times** and determine the *average results*

Row	UserID	MovieID	Rating
1	234	110	5
2	234	151	5
3	234	260	3
4	234	376	5
5	234	539	4 ^a
6	234	590	5
7	234	649	1
8	234	719	5 ^a
9	234	734	3
10	234	736	2

Note: 'a' = randomly selected ratings for testing.

	Training Set				Test Set	Measure e.g., RMSE
Round 1	r1, r2	r4, r7	r3, r8	r6, r10	r5, r9	$\text{RMSE}_{\text{round1}} = 0.5$
Round 2	r4, r7	r3, r8	r6, r10	r5, r9	r1, r2	$\text{RMSE}_{\text{round2}} = 0.2$
Round 3	r3, r8	r6, r10	r5, r9	r1, r2	r4, r7	$\text{RMSE}_{\text{round3}} = 0.1$
Round 4	r6, r10	r5, r9	r1, r2	r4, r7	r3, r8	$\text{RMSE}_{\text{round4}} = 0.2$
Round 5	r5, r9	r1, r2	r4, r7	r3, r8	r6, r10	$\text{RMSE}_{\text{round5}} = 0.1$
						Average RMSE = 0.22

Evaluation Tasks

- Prediction Task
 - Compute a *missing rating* in the user/item matrix.
 - Require Likert scale ratings.
- Classification Task
 - Select *a ranked list of n items relevant* for the user.
 - *Transform* Likert scale ratings into *relevant/irrelevant items*.

Metrics

- Accuracy of predictions
- Accuracy of classifications
- Accuracy of ranks

Accuracy of Predictions

- Mean absolute error

$$MAE = \frac{\sum_{u \in U} \sum_{i \in testset_u} |rec(u, i) - r_{u,i}|}{\sum_{u \in U} |testset_u|}$$

- Normalized MAE

$$NMAE = \frac{MAE}{r_{max} - r_{min}}$$

Accuracy of Classifications (1)

- Precision (= TP / (TP + FP))

$$P_u = \frac{|hits_u|}{|recset_u|}$$

$|hitS_u|$ = the number of correctly recommended relevant items for user u

- Recall (= TP / (TP + FN))

$$R_u = \frac{|hits_u|}{|testset_u|}$$

Example

- **Top-20** suggested items for user 234 ((Item,Rating) tuples):

$\text{Recset}_{234} = \{$

 $(912, 4.8),$
 $(47, 4.5),$
 $(263, 4.4),$
 $(\mathbf{539}, 4.1),$
 $(348, 4),$
 $\dots,$
 $(\mathbf{719}, 3.8) \}$

Top 3
Top 5

A Test Set

Row	UserID	MovieID	Rating
1	234	539	4
2	234	719	5

- $\text{MAE} = [|4.1 - 4| + |3.8 - 5|] / 2$
 $= (0.1 + 1.2) / 2 = 0.65$

- **Top 3** ranked items:

- $P_{234} = 0/3 = 0$
- $R_{234} = 0/2 = 0$

- **Top 5** ranked items:

- $P_{234} = 1/5 = 0.2$
- $R_{234} = 1/2 = 0.5$

- P_{234} and R_{234} for **Top 20** ranked items?

Accuracy of Classifications (2)

- F1 (F-measure)

$$F1 = \frac{2 \times P \times R}{P + R}$$

- Hit rate (focus on # of users who get at least 1 matched item)

$$hitrate_u = \begin{cases} 1 & \text{if } hit_u > 0 \\ 0 & \text{otherwise} \end{cases}$$

where,

hit_u = # of correctly recommended relevant items for user u

Accuracy of Ranks (*rankscore_u*')

$$rankscore_u = \sum_{i \in hit_u} \frac{1}{2^{\frac{rank(i)-1}{\alpha}}}$$

$$rankscore'_u = \frac{rankscore_u}{rankscore_u^{max}}$$

$$rankscore_u^{max} = \sum_{i \in testset_u} \frac{1}{2^{\frac{idx(i)-1}{\alpha}}}$$

where,

$rank(i)$ = the *position* of item i in the user's recommendation list

α = the *half-life* of utilities (decrease by half for each rank)

$idx(i)$ = assigning a value $\in \{1, \dots, |testset_u|\}$ corresponding to the order of $rank(i)$

Example

- Assume, $\alpha = 10$

20 (ItemID, Rating) tuples suggested for user 234:

$\text{Recset}_{234} = \{$
 (912, 4.8),
 (47, 4.5),
 (263, 4.4),
 (539, 4.1),
 (348, 4),
 ...,
 (719, 3.8) $\}$

Rank #4

Rank #20

$$\text{rankscore}'_u = \frac{\text{rankscore}_u}{\text{rankscore}_u^{\max}}$$

$$\text{rankscore}_u = \sum_{i \in \text{hit}_u} \frac{1}{2^{\frac{\text{rank}(i)-1}{\alpha}}}$$

$$\text{rankscore}_{234} = \frac{1}{2^{\frac{4-1}{10}}} + \frac{1}{2^{\frac{20-1}{10}}} = 1.08$$

$$\text{rankscore}'_{234} = \frac{1.08}{1.93} = 0.56$$

$$\text{rankscore}_u^{\max} = \sum_{i \in \text{testset}_u} \frac{1}{2^{\frac{\text{idx}(i)-1}{\alpha}}}$$

$$\text{rankscore}_{234}^{\max} = \frac{1}{2^{\frac{1-1}{10}}} + \frac{1}{2^{\frac{2-1}{10}}} = 1.93$$

Accuracy of Ranks (*liftindex_u*)

- Assume that the **ranked list** is *divided into ten equal deciles*.
- Count the number of hits in each decile as $S_{1,u}, S_{2,u}, \dots, S_{10,u}$, where

$$\sum_{i=1}^{10} S_i = \text{hits}_u$$

$$\text{liftindex}_u = \begin{cases} \frac{1 \cdot S_{1,u} + 0.9 \cdot S_{2,u} + \dots + 0.1 \cdot S_{10,u}}{\sum_{i=1}^{10} S_{i,u}} & \text{if } \text{hit}_u > 0 \\ 0 & \text{otherwise} \end{cases}$$

where,

hit_u = # of correctly recommended relevant items for user u

Example

20 Item/Rating tuples suggested for user 234:

$\text{Recset}_{234} = \{$

(912, 4.8),

(47, 4.5),

(263, 4.4),

(539, 4.1),

(348, 4),

$S_{1,u}$

$S_{2,u}$

...

...

(719, 3.8)

$S_{10,u}$

$$\text{liftindex}_u = \begin{cases} \frac{1 \cdot S_{1,u} + 0.9 \cdot S_{2,u} + \dots + 0.1 \cdot S_{10,u}}{\sum_{i=1}^{10} S_{i,u}} & \text{if } \text{hit}_u > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\text{liftindex}_{234} = \frac{\overset{S_{2,u}}{0.9 \cdot \boxed{1}} + \overset{S_{10,u}}{0.1 \cdot \boxed{1}}}{2} = 0.5$$

Discounted Cumulative Gain (DCG)

- **Assumption:** The ***highly relevant*** documents are ***more useful*** than moderately relevant documents.
- DCG measures ***ranking quality*** that assesses the recommended list provided by a recommendation engine

$$DCG = \sum_{i=1}^n \frac{\text{relevance}_i}{\log_2(i+1)}$$

Document relevant score of the ordered recommended item_i

- E.g., score_reset_A = [2,3,2,3,1] and score_reset_B = [3,3,2,1,2]

$$DCG_A = \frac{2}{\log_2(1+1)} + \frac{3}{\log_2(2+1)} + \frac{2}{\log_2(3+1)} + \frac{3}{\log_2(4+1)} + \frac{1}{\log_2(5+1)} = 6.57$$

$$DCG_B = \frac{3}{\log_2(1+1)} + \frac{3}{\log_2(2+1)} + \frac{2}{\log_2(3+1)} + \frac{1}{\log_2(4+1)} + \frac{2}{\log_2(5+1)} = 7.09$$

Normalized Discounted Cumulative Gain (NDCG)

$$NDCG = \frac{DCG}{iDCG}$$

- where, iDCG = DCG of the ideal order
- $NDCG \in [0, 1]$
- E.g., $score_recset_A = [2, 3, 2, 3, 1]$ and $ideal_score_recset_A = [3, 3, 2, 2, 1]$

$$DCG_A = \frac{2}{\log_2(1+1)} + \frac{3}{\log_2(2+1)} + \frac{2}{\log_2(3+1)} + \frac{3}{\log_2(4+1)} + \frac{1}{\log_2(5+1)} = 6.57$$

$$iDCG_A = \frac{3}{\log_2(1+1)} + \frac{3}{\log_2(2+1)} + \frac{2}{\log_2(3+1)} + \frac{2}{\log_2(4+1)} + \frac{1}{\log_2(5+1)} = 7.14$$

$$NDCG = \frac{6.57}{7.14} = 0.92$$

Note: use **Mean NDCG** of all test users' recommended list to evaluate the performance of the recommendation algorithm