KSE801 – Recommender System and Machine Learning on Graph

Lecture 2: Neighborhood-Based Collaborative Filtering

Prof. Chanyoung Park

Department of Industrial & Systems Engineering
KAIST
cy.park@kaist.ac.kr

RECALL: BASIC MODELS OF RECOMMENDATION

- **Collaborative filtering (CF)** – **Only ratings**
  - Memory-based approach (Neighborhood-based CF) **(Week 2)**
  - Model-based approach **(Week 3 ~ 4)**
- **Side information-based Recommendation (Week 5 ~ 6)**
  - Content-based recommendation – **Only contents (Week 1)**
  - Content-based CF – **Rating + Contents**
    - Text, image, social network, etc
- **Advanced topics**
  - Sequential Recommendation & Graph-based Recommendation **(Week 7)**

OUTLINE

- Neighborhood-based Collaborative Filtering
- A Regression Modeling View of Neighborhood Methods
- Evaluating Recommender System

OUTLINE

- Neighborhood-based Collaborative Filtering
- A Regression Modeling View of Neighborhood Methods
- Evaluating Recommender System

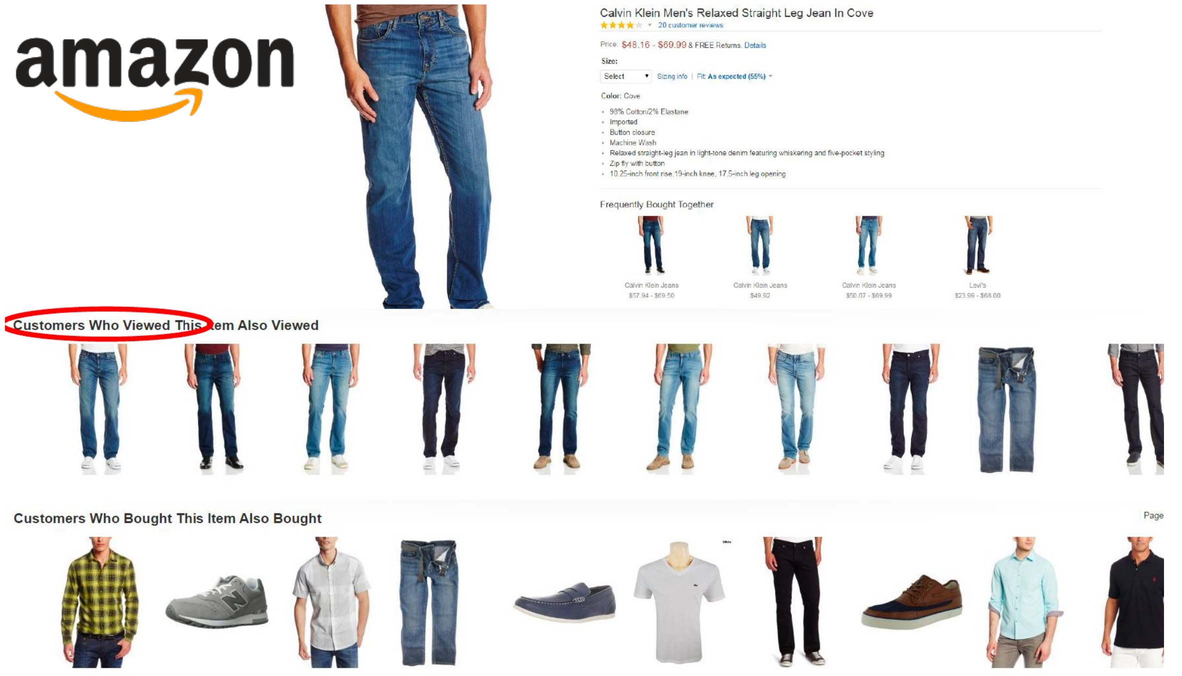
COLLABORATIVE FILTERING (CF)

- **The most prominent approach to generate recommendations**
  - Used by large, commercial e-commerce sites
  - Well-understood, various algorithms and variations exist
  - Applicable in many domains (book, movies, DVDs, ..)

- **Approach**
  - Use the "wisdom of the crowd" to recommend items
- **Basic assumption and idea**
  - Users give ratings to items (implicitly or explicitly)
  - **Customers who had similar tastes in the past, will have similar tastes in the future**

NEIGHBORHOOD-BASED CF: INTRODUCTION

- Among the earliest algorithms developed for collaborative filtering
- **Main idea**
  - **Similar users display similar patterns of rating behavior** → **User-based CF**
  - **Similar items receive similar ratings** → **Item-based CF**



- **How do we define similarity between users and items?**
  - **Q.** How can we measure the **similarity** between two **users?**
    - **A:** In terms of the **items** they purchased!
  - **Q:** How can we measure the **similarity** between two **items?**
    - **A:** In terms of the **users** who purchased them!

INPUT AND OUTPUT OF CF

- **Input**
  - Only a matrix of given user–item ratings
- **Output types**
  - A (numerical) prediction indicating to what degree the current user will like or dislike a certain item
    - Explicit feedback: Rating prediction
  - A top-N list of recommended items
    - Implicit feedback: Ranking

USER-BASED COLLABORATIVE FILTERING

(SIMILARLY DEFINED FOR ITEM-BASED CF)

- Given an "active user" (Alice) and an item i not yet seen by Alice,
  - find a set of users who liked the same items as Alice in the past **and** who have rated item i
  - use, e.g. the average of their ratings to predict, if Alice will like item i
  - do this for all items Alice has not seen and recommend the best-rated
- **Basic assumption and idea**
  - If users had similar tastes in the past they will have similar tastes in the future
  - User preferences remain stable and consistent over time
- **Example:** Determine whether Alice will like or dislike "*Item 5*", which Alice has not yet rated or seen

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

USER-BASED COLLABORATIVE FILTERING

(SIMILARLY DEFINED FOR ITEM-BASED CF)

- **Some first questions**
  - How do we measure similarity?
  - How many neighbors should we consider?
  - How do we generate a prediction from the neighbors' ratings?

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Explicit feedback

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

Implicit feedback

NOTATIONS

- Definitions
  - I_u = Set of items purchased by user u
  - U_i = Set of users who purchased item i

$$R = \begin{array}{cc} \text{Items} & \\ \begin{pmatrix} 1 & 0 & \cdots & 1 \\ 0 & 0 & & 1 \\ \vdots & & \ddots & \vdots \\ 1 & 0 & \cdots & 1 \end{pmatrix} & \text{Users} \end{array}$$

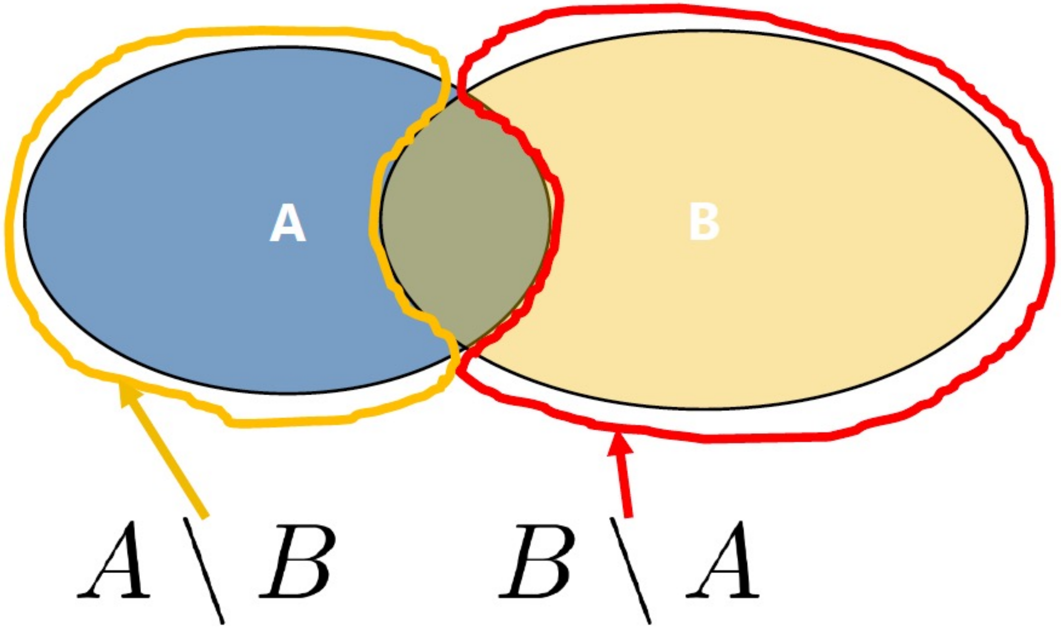
R_u: Binary representation of items purchased by user u

$R_{:,i}$: Binary representation of users who purchased item i

$$I_u = \{i | R_{u,i} = 1\}, U_i = \{u | R_{u,i} = 1\}$$

MEASURING USER SIMILARITY: IMPLICIT FEEDBACK

- **1) Euclidean distance**
  - Between two items i, j or Between two users u, v



Euclidean distance $(A, B) = |A \setminus B| + |B \setminus A|$

$$\left|U_i \setminus U_j\right| + \left|U_j \setminus U_i\right|$$

Example 1

$U_1 = \{1,4,8,9,11,23,25,34\}$

$U_2 = \{1,4,6,8,9,11,23,25,34,35,38\}$

$|U_1 \setminus U_2| + |U_2 \setminus U_1| = 3$

Example 2

$U_3 = \{4\}$

$U_4 = \{5\}$

$|U_3 \setminus U_4| + |U_4 \setminus U_3| = 2$

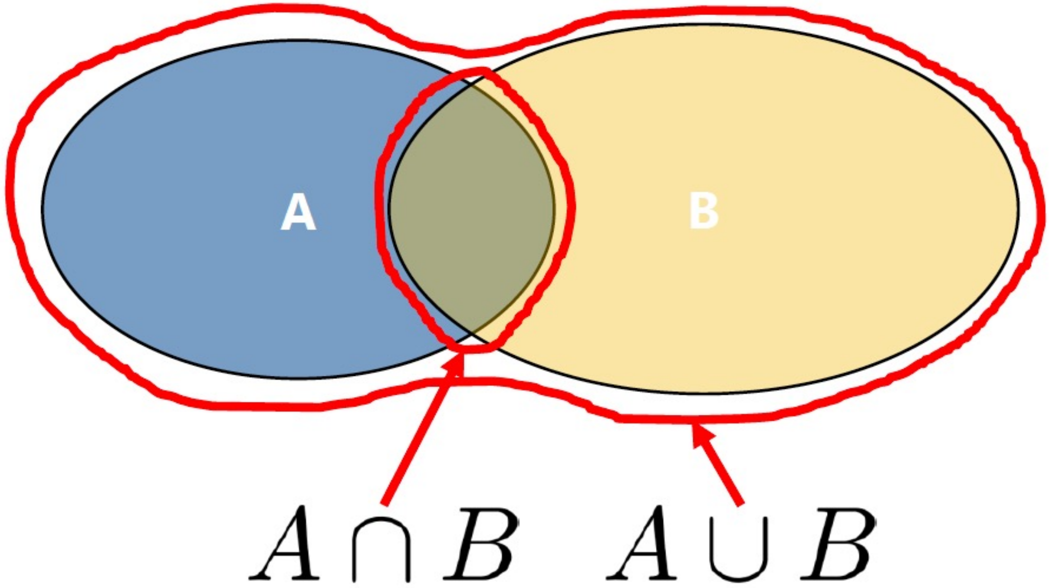
Problem: Euclidean distance favors small sets, even if they have few elements in common

MEASURING USER SIMILARITY: IMPLICIT FEEDBACK

- **2) Jaccard similarity**
  - Between two items i, j or Between two users u, v

$$\text{Jaccard}(U_i, U_j) = \frac{|U_i \cap U_j|}{|U_i \cup U_j|}$$

$$(0 \leq \text{Jaccard}(U_i, U_j) \leq 1)$$



$$\text{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Maximum of 1 if the two users purchased exactly the same set of items. i.e., $U_i = U_j$

Minimum of 0 if the two users purchased completely disjoint sets of items. i.e., $U_i \cap U_j = \emptyset$

Problem: Jaccard similarity can only take care of binary relevance scores

MEASURING USER SIMILARITY: IMPLICIT FEEDBACK

Cosine Similarity $(A,B) = \dfrac{A \cdot B}{\|A\|\|B\|}$

- **3) Cosine similarity**
  - Between two items i, j or Between two users u, v
  - Unlike Jaccard, works for arbitrary vectors

Purchase history $= \begin{pmatrix} 1 & 0 & \cdots & 1 \\ 0 & 0 & & 1 \\ \vdots & & \ddots & \vdots \\ 1 & 0 & \cdots & 1 \end{pmatrix} \Rightarrow \begin{pmatrix} -1 & 0 & \cdots & 1 \\ 0 & 0 & & -1 \\ \vdots & & \ddots & \vdots \\ 1 & 0 & \cdots & -1 \end{pmatrix}$

Didn't buy **Bought and liked**

Bought and hated **Bought and liked** **Didn't buy**

If cosine$(I_u, I_v) = 1$?

→ Angle between $I_u, I_v = 0$

→ Users u and v rated the same items and they all agree

If cosine$(I_u, I_v) = -1$?

→ Angle between $I_u, I_v = 180$

→ Users u and v rated the same items and they all disagree

If cosine$(I_u, I_v) = 0$?

→ Angle between $I_u, I_v = 90$

→ Users u and v rated completely different sets of items

What if we have numerical ratings?

MEASURING USER SIMILARITY: EXPLICIT FEEDBACK

- A popular similarity measure in user-based CF: **Pearson correlation**
  - Possible similarity values between -1 and 1

- **Different users may provide ratings on different scales**
  - **Solution:** Subtract the average
    - Values are negative for below-average ratings and positive for above-average ratings

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

sim = 0.85

sim = 0.00

sim = 0.70

sim = -0.79

Items rated by both users u and v

Average rating by user v

$$Pearson(u,v) = \frac{\sum_{i \in I_u \cap I_v} (R_{u,i} - \bar{R}_u)(R_{v,i} - \bar{R}_v)}{\sqrt{\sum_{i \in I_u \cap I_v} (R_{u,i} - R_u)^2 \sum_{i \in I_u \cap I_v} (R_{v,i} - \bar{R}_v)^2}}$$

MAKING PREDICTIONS

- **A common prediction function**
  - **User-based CF:** The most similar items should be the most relevant when predicting future ratings, so the user's past ratings of those items are given the highest weights

$$\hat{R}_{u,i} = \frac{\sum_{v \in U_i} sim(u,v) * R_{v,i}}{\sum_{v \in U_i} sim(u,v)} \Rightarrow \hat{R}_{u,i} = \bar{R}_u + \frac{\sum_{v \in U_i} sim(u,v) * (R_{v,i} - \bar{R}_v)}{\sum_{v \in U_i} sim(u,v)}$$

  - **Item-based CF**

$$\hat{R}_{u,i} = \frac{\sum_{j \in I_u} sim(i,j) * R_{u,j}}{\sum_{j \in I_u} sim(i,j)} \Rightarrow \hat{R}_{u,i} = \bar{R}_{:,i} + \frac{\sum_{j \in I_u} sim(i,j) * (R_{u,j} - \bar{R}_{:,j})}{\sum_{j \in I_u} sim(i,j)}$$

- **How many neighbors, i.e.,$|U_i|$, $|I_u|$?**
  - Only consider positively correlated neighbors (or higher threshold)
  - Can be optimized based on data set (e.g., Cross-validation)
  - Often, between 50 and 200

USER-BASED CF vs. ITEM-BASED CF

- **Item-based methods** often provide more relevant recommendations because **a user's own ratings are used** to perform the recommendation
  - i.e., Similar items are those that are consumed by the user, and the user's own ratings on those items are used to extrapolate the ratings of the target item
- Item-based methods can also provide a **reason for the recommendation**

Because you watched "Secrets of the Wings," [the recommendations are…]

- For user-based methods, "neighboring users" are **simply a set of anonymous users**
- In social recommendation setting, actual friends can be "neighboring users", which can provide explanations for the recommendation (Later in this course)

IMPROVING THE METRICS

- **Not all neighbor ratings might be equally "valuable"**
  - Agreement on commonly liked items is not so informative as agreement on controversial items
  - **Possible solution**: Give more weight to items that have a higher variance
- **Value of number of co-rated items**
  - Use “significance weighting”, by e.g., linearly reducing the weight when the number of co-rated items is low
- **Case amplification**
  - Intuition: Give more weight to "very similar" neighbors, i.e., where the similarity value is close to 1.
- **Neighborhood selection**
  - Use similarity threshold or fixed number of neighbors

NEIGHBORHOOD-BASED CF: PROS/CONS

- Neighborhood-based CF is also known as **memory-based CF** (Why?)
  - vs. Model-based CF
- **Pros**
  - Requires minimal knowledge (only rating/implicit feedback)
  - Produces good-enough results in most cases
- **Challenges**
  - **Sparsity / Cold-start problem**
    - We need enough users in the system
    - New items need to get enough ratings.
    - New users need to provide enough ratings (cold start)
  - **Scalability**
    - Nearest neighbor require computation that grows with both the number of users and the number of items
    - If one user purchases one item, this will change the rankings of **every other item that was purchased by at least one user in common**
  - **Diversity:** Does not encourage diverse results

SPARSITY / COLD-START PROBLEM

- How to recommend new items? What to recommend to new users?

- **New User Problem**: the system must first learn the user's preferences from the ratings
  - E.g., Watcha asks new users to rate 10 movies when they register
  - Hybrid RS, which combines content-based and collaborative techniques, can help

- **New Item Problem**: Until the new item is rated by a substantial number of users, the RS is not able to recommend it
  - Hundreds/thousands of new items every day
    - Yahoo News: ~100 new articles / day
    - eBay or Amazon: >1000 items / day ???

WATCHA

OUTLINE

- Neighborhood-based Collaborative Filtering
- A Regression Modeling View of Neighborhood Methods
- Evaluating Recommender System

A REGRESSION MODELING VIEW OF NEIGHBORHOOD METHODS

- Recall the prediction function of item-based CF

$$\hat{R}_{u,i} = \bar{R}_{:,i} + \frac{\sum_{j \in I_u} sim(i,j) * (R_{u,j} - \bar{R}_{:,j})}{\sum_{j \in I_u} sim(i,j)} \qquad \begin{array}{c} \forall u \in \{1 \dots m\} \\ \forall i \in \{1 \dots n\} \end{array}$$

- The predicted rating $\hat{R}_{u,i}$ is a **weighted linear combination** of other ratings of the same user u, i.e., I_u
- If we allow I_u to contain all items, then the function is similar to linear regression
  - Linear regression: The coefficients are determined by optimization model
  - This case: The coefficients are **heuristically** determined by $sim(i,j)$
- Can we think about an optimization-based neighborhood methods?

$$W^{item} \in R^{n \times n}$$

$$\hat{R}_{u,i} = \bar{R}_{:,i} + \frac{\sum_{j \in I_u} sim(i,j) * (R_{u,j} - \bar{R}_{:,j})}{\sum_{j \in I_u} sim(i,j)} \Rightarrow \hat{R}_{u,i} = \bar{R}_{:,i} + \sum_{j \in I_u} W_{i,j}^{item} * (R_{u,j} - \bar{R}_{:,j})$$

SPARSE LINEAR MODELS (SLIM)

- SLIM **does not restrict the regression coefficients to only the neighborhood** of the target item

$$\hat{R}_{u,i} = \sum_{j \in I_u} W_{i,j}^{item} * R_{u,j} \Rightarrow \hat{R}_{u,i} = \sum_{j=1}^{n} W_{i,j}^{item} * R_{u,j} \Rightarrow \hat{R} = RW^{item}$$

- The target item itself is excluded on the right-hand side to prevent overfitting
  - This can be achieved by $W_{t,t}^{item} = 0$, for $t = 1, \dots, n$

Elastic-net regularizer

$$\min_{W} \frac{1}{2} \left\| R - RW^{item} \right\|_F^2 + \frac{\beta}{2} \left\| W^{item} \right\|^2 + \lambda \left| W^{item} \right|$$

subject to $W^{item} \geq 0$

$diag\left(W^{item}\right) = 0$

$$|W| = \sum_{i=1}^{n} \sum_{j=1}^{n} |W_{i,j}|$$

$$\|W\|^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}^2$$

X. Ning and G. Karypis. SLIM: Sparse linear methods for top-N recommender systems. ICDM, 2011.

RECALL: SHRINKAGE METHODS (IN LINEAR REGRESSION)

- Shrinkage methods *constrain* or *regularize* the coefficient estimates, or equivalently shrink the coefficient es timates to zero
- It turns out that shrinking estimated coefficients towards zero can significantly reduce their variance
- Two best-known techniques are **ridge regression** and **lasso**

- **Overview**
  - Linear regression

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{argmin} \|\boldsymbol{y} - \boldsymbol{X\beta}\|^2$$

  - Ridge regression

L2 norm

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{argmin} (\|\boldsymbol{y} - \boldsymbol{X\beta}\|^2 + \lambda \|\boldsymbol{\beta}\|^2), \text{ where } \|\boldsymbol{\beta}\|^{\mathbf{2}} = \sum_{j=1}^{p} \beta_j^2$$

  - Lasso

L1 norm

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{argmin} (\|\boldsymbol{y} - \boldsymbol{X\beta}\|^2 + \lambda |\boldsymbol{\beta}|), \text{ where } |\boldsymbol{\beta}| = \sum_{j=1}^{p} |\beta_j|$$

RECALL: RIDGE vs. LASSO

- The Lasso and ridge regression coefficient estimates solve the following problems
- For each value of λ, there exists a value for s such that
  - Ridge regression

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j X_{ij}\right)^2 \quad \text{subject to} \quad \sum_{j=1}^{p}\beta_j^2 \le s$$

  - Lasso

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j X_{ij}\right)^2 \quad \text{subject to} \quad \sum_{j=1}^{p}|\beta_j| \le s$$

THE LASSO PICTURE: COMPARING CONSTRAINT FUNCTIONS

$$RSS = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j X_{ij}\right)^2$$

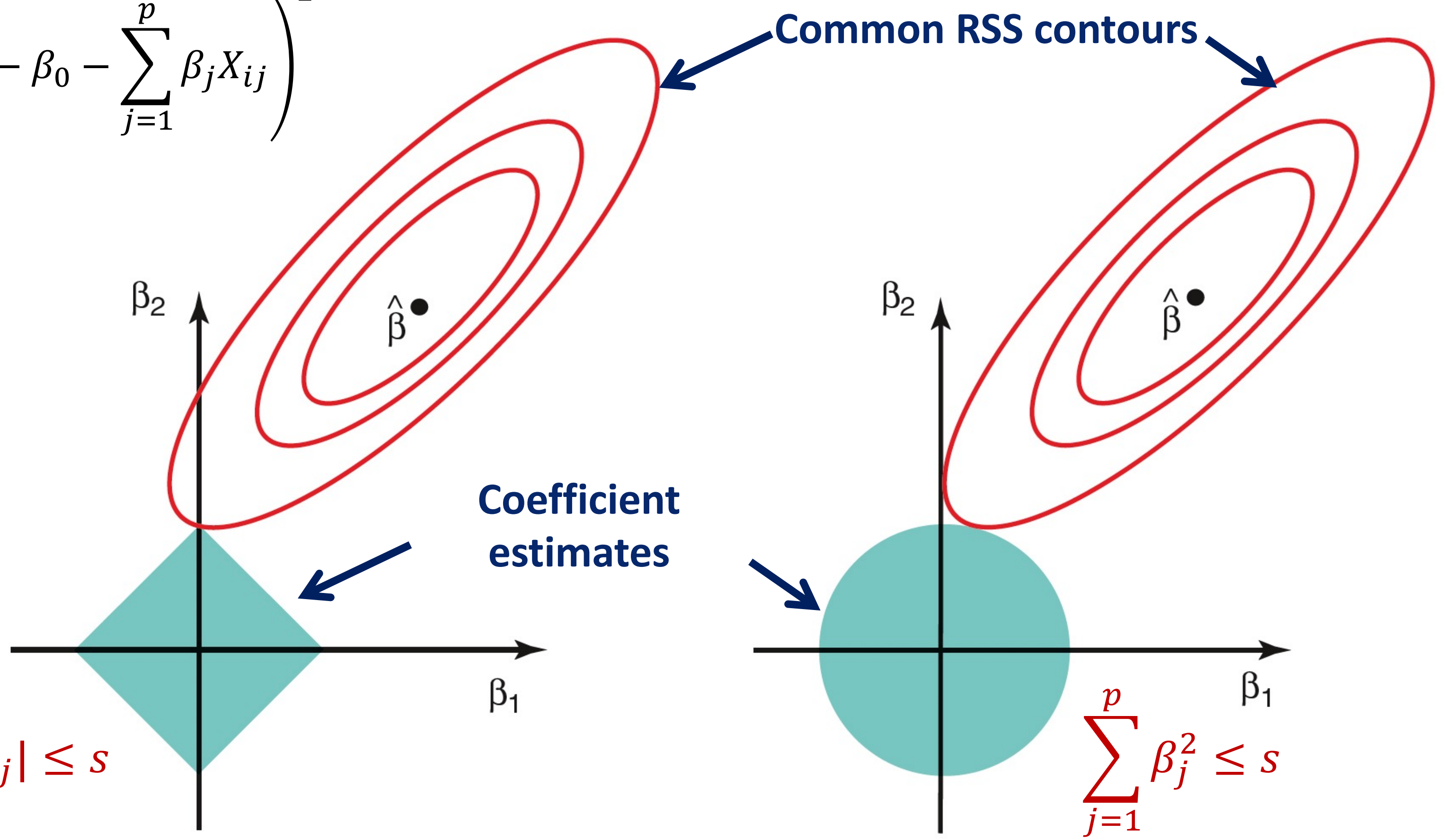


FIGURE 6.7. *Contours of the error and constraint functions for the lasso* (left) *and ridge regression* (right). *The solid blue areas are the constraint regions,* $|\beta_1| + |\beta_2| \leq s$ *and* $\beta_1^2 + \beta_2^2 \leq s$, *while the red ellipses are the contours of the RSS.*

OUTLINE

- Neighborhood-based Collaborative Filtering
- A Regression Modeling View of Neighborhood Methods
- Evaluating Recommender System

EVALUATING RECOMMENDER SYSTEM

- Recall, there are broadly two tasks in recommender system
- We need different metrics for different tasks

- **1) Rating prediction → Regression**
  - **Input**: set of ratings for user/item pairs
  - **Output**: map from user/item pair to predicted rating
  - Metrics: Root Mean Square Error (RMSE), Mean Absolute Error (MAE)

- **2) Item ranking (top-k recommendation) → Ranking**
  - **Input**: set of user/item pairs such as shopping data, and an integer k
  - **Output**: a list of k items for each user which are most likely to be bought by him/her
  - Metrics: Precision, Recall, ROC/AUC, F1, AP, RR, NDCG

	GLADITOR	GODFTHER	BEN-HUR	GOODFELLAS	SCARFACE	SPARTACUS
U_1	1			5		2
U_2		5			4	
U_3	5	3		1		
U_4			3			4
U_5				3	5	
U_6	5		4			?

(a) Ordered ratings

	GLADIATOR	GODFATHER	BEN-HUR	GOODFELLAS	SCARFACE	SPARTACUS
U_1	1			1		1
U_2		1			1	
U_3	1	1		1		
U_4			1			1
U_5				1	1	
U_6	1		1			?

(b) Unary ratings

EVALUATION OF RATING PREDICTION

- **Datasets with items rated by users**
  - MovieLens datasets 100K-10M ratings
  - Netflix 100M ratings
- **Metrics measure error rate**
  - **Mean Absolute Error (*MAE*):** Computes the deviation between predicted ratings and actual ratings

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

  - **Root Mean Square Error (*RMSE*):** Similar to *MAE*, but places more emphasis on larger deviation

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

EVALUATION OF ITEM RANKING

		Reality	
		Actually Good	**Actually Bad**
Prediction	**Rated Good**	True Positive (tp)	False Positive (fp)
	Rated Bad	False Negative (fn)	True Negative (tn)

Confusion matrix

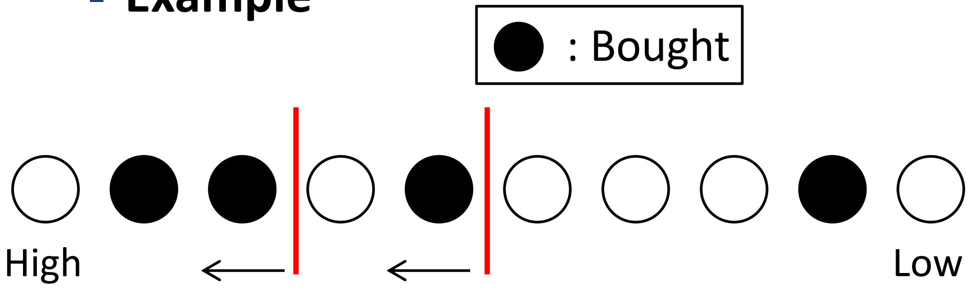
- **Precision**
  - A measure of exactness, determines the fraction of relevant items retrieved out of all items retrieved
  - E.g. the proportion of recommended movies that are actually good

$$Precision = \frac{tp}{tp + fp} = \frac{|good\ movies\ recommended|}{|\text{all recommendations}|}$$

- **Recall**
  - A measure of completeness, determines the fraction of relevant items retrieved out of all relevant items
  - E.g. the proportion of all good movies recommended

$$Recall = \frac{tp}{tp + fn} = \frac{|good\ movies\ recommended|}{|all\ good\ movies|}$$

- **Example**

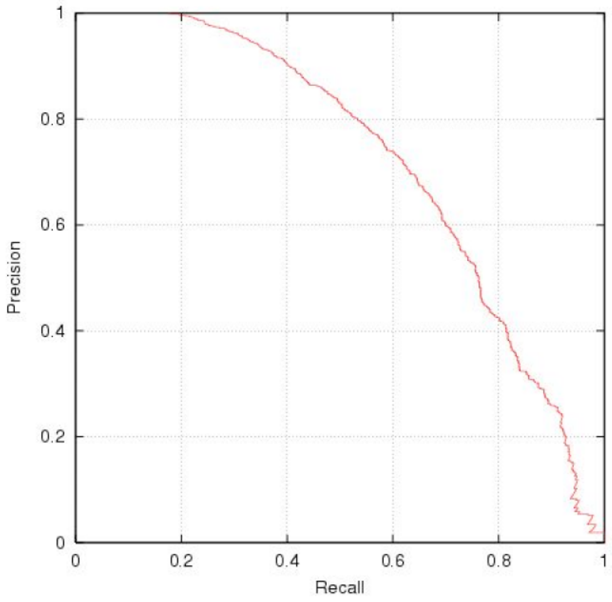


Assuming that 3 items are recommended:

- 2 out of 3 recommended items are actually bought: Precision@3=2/3
- 2 out of 4 bought items are recommended: Recall@3=2/4
- Ex) Precision@5 = 3/5, Recall@5 = 3/4

TRADEOFF BETWEEN PRECISION AND RECALL

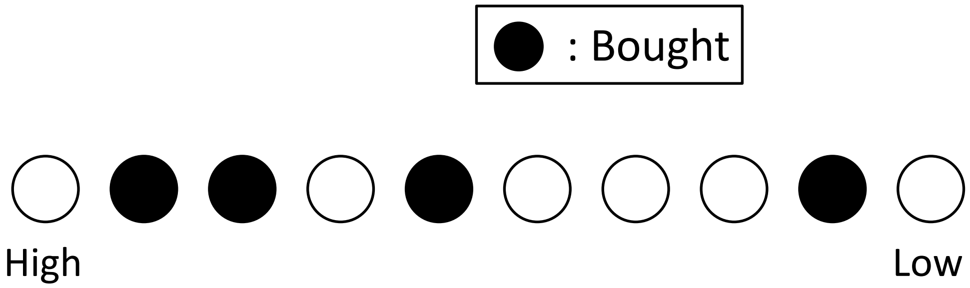
- Typically when a recommender system is tuned to increase precision, recall decreases as a result (or vice versa)



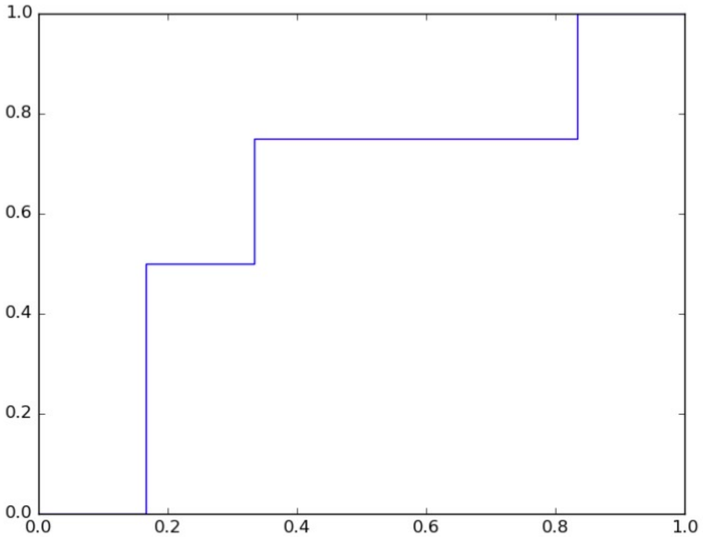
- The **F_1 Metric** attempts to combine Precision and Recall into a single value for comparison purposes.
  - May be used to gain a more balanced view of performance

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

ROC & AUC



Num. recommendation	1	2	3	4	5	6	7	8	9	10
Num. whites	1	1	1	2	2	3	4	5	5	6
Num. blacks	0	1	2	2	3	3	3	3	4	4



- Divide the first and second row by total number of white and blacks respectively, and plot the values
- This curve is called **ROC curve**
- The area under this curve is called **AUC**
- Higher AUC is better (max =1)

RANK POSITION MATTERS

- For a user:

Actually good
Item 237
Item 899

hit

Recommended (predicted as good)
Item 345
Item 237
Item 187

- **Rank metrics** extend recall and precision to take the **positions of correct items** in a ranked list into account
  - Relevant items are more useful when they appear earlier in the recommendation list
  - Particularly important in recommender systems as lower ranked items may be overlooked by users
  - Average precision (AP), Reciprocal rank (RR), Normalized discounted cumulative gain (NDCG)

https://medium.com/swlh/rank-aware-recsys-evaluation-metrics-5191bba16832

AVERAGE PRECISION

- **Average Precision** (*AP*) is a ranked precision metric that places emphasis on highly ranked correct predictions (hits)

- Example

Rank	Hit?
1	
2	✓
3	✓
4	✓
5	

$$AP = \frac{1}{3}\left(\frac{1}{2} + \frac{2}{3} + \frac{3}{4}\right) = \frac{23}{36} \approx 0.639$$

Rank	Hit?
1	✓
2	
3	
4	✓
5	✓

$$AP = \frac{1}{3}\left(\frac{1}{1} + \frac{2}{4} + \frac{3}{5}\right) = \frac{21}{30} = 0.7$$

$$mAP = \frac{1}{|U|}\sum_{i=1}^{|U|} AP(u)$$

$$AP(u) = \frac{1}{|S|}\sum_{k=1}^{n} prec@k \cdot rel(k)$$

RECIPROCAL RANK

- **Reciprocal Rank** (RR) is the sum of the inverse of the rank of the relevant items (hit) in a given list.

- Example

Rank	Hit?
1	
2	X
3	X
4	X
5	

$$RR = \frac{1}{2} + \frac{1}{3} + \frac{1}{4}$$

Rank	Hit?
1	X
2	
3	
4	X
5	X

$$RR = 1 + \frac{1}{4} + \frac{1}{5}$$

$$MRR = \frac{1}{|U|} \sum_{u=1}^{|U|} RR(u)$$

$$RR(u) = \sum_{i=1}^{k} \frac{relevance_i}{rank_i}$$

NORMALIZED DISCOUNTED CUMULATIVE GAIN (NDCG)

- **Discounted cumulative gain (DCG)**
  - Logarithmic reduction factor

$$DCG_{pos} = rel_1 + \sum_{i=2}^{pos} \frac{rel_i}{\log_2 i}$$

  - pos denotes the position up to which relevance is accumulated
  - rel_i returns the relevance of recommendation at position i

Rank	Hit?
1	
2	X
3	X
4	X
5	

- **Idealized discounted cumulative gain (IDCG)**
  - Assumption that items are ordered by decreasing relevance

$$IDCG_{pos} = rel_1 + \sum_{i=2}^{|h|-1} \frac{rel_i}{\log_2 i}$$

$$DCG_5 = \frac{1}{\log_2 2} + \frac{1}{\log_2 3} + \frac{1}{\log_2 4} = 2.13$$

$$IDCG_5 = 1 + \frac{1}{\log_2 2} + \frac{1}{\log_2 3} = 2.63$$

- **Normalized discounted cumulative gain (nDCG)**
  - Normalized to the interval [0..1]

$$nDCG_{pos} = \frac{DCG_{pos}}{IDCG_{pos}}$$

$$nDCG_5 = \frac{DCG_5}{IDCG_5} \approx 0.81$$

CONCLUSION

- Neighborhood-based Collaborative Filtering
  - User-based CF / Item-based CF
  - Measuring similarity
  - Pros and cons of neighborhood-based CF
  - Sparsity / Cold-start problem

- A Regression Modeling View of Neighborhood Methods
  - Sparse Linear Model (SLIM)
- Evaluating Recommender System
  - Rating prediction and item ranking

Coming up next:

Model-based CF