Recommendation Systems

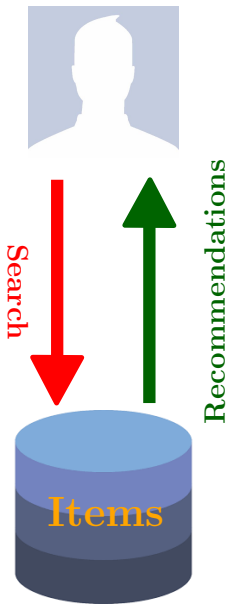
- Recommenders: Motivation and Applications
- Problem Formulation and Evaluation
- Methods and Techniques (ANOVA and Averages)
- Content Based Filtering
- Collaborative Filtering
- Matrix Factorization
- Rank Factorization
- Low-Rank Structure and Singular Value Decomposition
- Hybrid Methods for Recommendation Systems

Recommenders: Motivation and Applications

Recommendation systems aim to provide users with personalized suggestions based on their preferences and behavior

CNN Money

> The Web, they say, is leaving the era of search and entering one of discovery. What's the difference? Search is what you do when you're looking for something. Discovery is when something wonderful that you didn't know existed, or didn't know how to ask for, finds you.
>
> J. O'Brien, Nov 20, 2006 The race to create a 'smart' Google

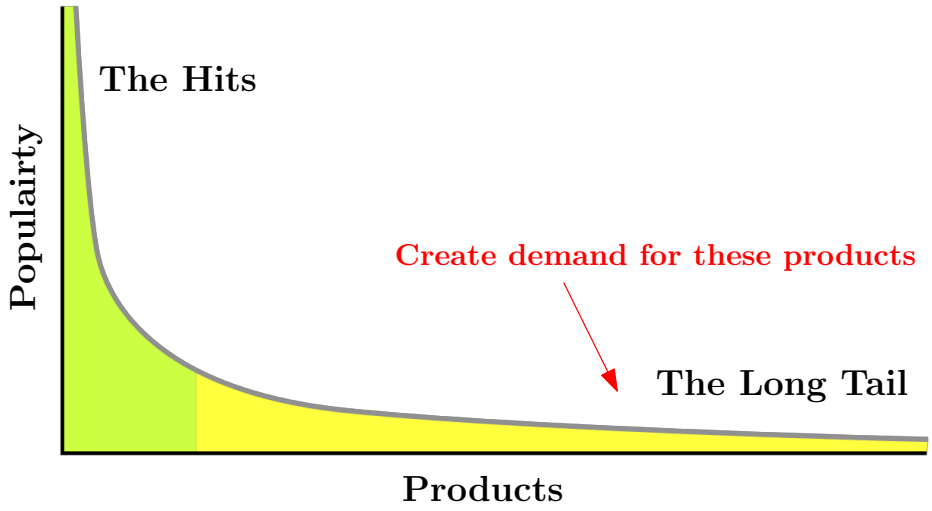


Products, news, friends, websites,movies, courses

- Early 1990s: using predefined rules to suggest items
  - **Hand-Curated:** Chef's specials, editor's picks, favorites
  - **Simple aggregates:** Top 10, Trending, Recent uploads
- Mid-1990s: Collaborative filtering approaches
  - Tapestry (1992): Allowing users to manually annotate documents and share recommendations
  - GroupLens (1994): Recommending Usenet news articles, using user-user similarity to predict preferences
- 2000s: Content-based and hybrid recommenders
- 2010s: Context-aware and personalized recommenders

Recommendation Systems

- Retailers cannot shelve everything
- Online retailers and digital content providers have millions of products



- Near zero-cost dissemination of information about products
- More choice necessitates better information filtering (customization)
- Data-driven recommendation customized to individual user

Recommendation Systems

Harvard Business Review 2017

Perhaps the single most important algorithmic distinction between "born digital"' enterprises and legacy companies is not their people, data sets, or computational resources, but a clear real-time commitment to delivering accurate, actionable customer recommendations.

- Netflix: 75% of movies watched are recommended [1]
  - "... personalization and recommendations save us more $\$1B$ per year" [2]
- Amazon: 35% of purchases on Amazon come from recommendations [1]
- Google News: recommendation genearate 38% more click-throughs [1]
- Airbnb: "Together, Search Ranking and Similar Listings drive 99% of our booking conversions" [3]
- Alibaba: For 11.11. mega sale, targeted personalized landing pages, resulted in 20% higher conversion rate from previous year [4]

[1] X. Amatriain, (2014) Machine Learning Summer School, CMU

[2] Gomez-Uribe & Hunt, Netflix Inc., (ACM Trans. on MIS 2015)

[3] Grbovic et.al [Airbnb Engineering & Data Science] (2018)

[4] InsideRetail.Asia (2017)

Growth in Importance Across Different Sectors

Recommendation systems have become critical in various sectors

- E-commerce: Driving sales by suggesting products
- Content Streaming: Content recommendations to keep users engaged
- Social Media: Suggest friends, pages, content to improve user retention

Recommendation Systems: Monetization of User Preferences

Monetizing user preferences through recommendation systems via

- Targeted Advertising: By analyzing user behavior, platforms can present ads tailored to individual preferences, increasing the likelihood of clicks and purchases
- Product Recommendations: E-commerce platforms like Amazon use collaborative filtering to suggest items frequently bought together or by similar users, boosting sales
- Subscription Retention: Media platforms like Netflix use recommendations to encourage users to discover new content, reducing the likelihood of subscription cancellations

Recommendation Systems: User Retention

Recommendation systems significantly enhance user experience and retention:

- Reduced Cognitive Load: Users no longer need to browse through large catalogs; recommendations provide them with relevant items quickly
- Engagement: Personalized recommendations lead to higher user engagement, as users spend more time interacting with suggested content
- Retention: Platforms that offer valuable recommendations foster user loyalty, as users feel the system understands their preferences and returns relevant content

Recommendation Systems: User Engagement

Social networks rely on recommendation systems to create personalized experiences for their users (Personalized Feeds):

- Friend Suggestions: Platforms like Facebook use collaborative filtering and deep learning to suggest friends based on mutual connections and similar interests
- Content and Page Recommendations: Platforms such as Twitter and Instagram suggest posts, pages, or groups based on a user's past interactions and behaviors
- Advertisement Personalization: Social networks utilize user data for personalized advertisements, increasing engagement and monetization

Facebook's recommenders are key drivers behind its friend suggestions, personalized content feeds, and targeted ads, significantly improving user engagement and time spent on the platform

Problem Formulation and Evaluation

Recommendation Systems: Problem Formulation

- n users - $\{c_1, \ldots, c_n\}$ and m items - $\{p_1, \ldots, p_m\}$
- Utility Matrix U: $n \times m$ matrix row/column for each user/item
- $U(i, j)$: rating of user i for item j

	p_1	p_2	p_3						p_j									p_m
u_1	1		2	1		4			2		3	2		5				2
u_2		1				2		1			2			1				3
u_3		1	1	2				1							1		2	
				3		2			5			2			3	4		
		1			2									5				
u_i			3	2	1		4	5		1		3	1		2			1
			4													4		
			5			1										5		
		1		4							1	3		5		1	2	
u_n			3		1	1		2	1					4				5

$U(i, j)$ could be

- $0 - 5$ stars
- $\in [0, 1]$
- $\in \{0, 1\}$

Computational linear algebra problem of **matrix completion**

Recommendation Systems: Problem Formulation

- n users - $\{c_1, \ldots, c_n\}$ and m items - $\{p_1, \ldots, p_m\}$
- Utility Matrix U: $n \times m$ matrix row/column for each user/item
- $U(i, j)$: rating of user i for item j

	p_1	p_2	p_3						p_j									p_m
u_1	1		2	1		4			2		3	2		5				2
u_2		1				2		1			2			1				3
u_3		1	1	2				1							1		2	
				3		2			5			2			3	4		
		1			2									5				
u_i			3	2	1		4	5	?	1		3	1		2			1
			4													4		
			5			1										5		
		1		4							1	3		5		1	2	
u_n			3		1	1		2	1					4				5

$U(i, j)$ could be

- $0 - 5$ stars
- $\in [0, 1]$
- $\in \{0, 1\}$

If prediction for $U(i, j)$ is high recommend product j to user i

Evaluation of Recommendation Systems

Understand its effectiveness and ensuring it meets user needs

- Measuring Effectiveness: Ensures that the recommendation system is providing relevant and useful suggestions
- Identifying Weaknesses: Helps uncover areas where the model is underperforming, such as handling new users or items
- Balancing Multiple Objectives: Evaluation helps measure trade-offs between different objectives like accuracy, diversity, and novelty
- User Satisfaction: Directly impacts the user experience by ensuring recommendations are aligned with user preferences
- Business Objectives: Evaluating the system helps determine if the recommendations are driving engagement, retention, and revenue
- Model Comparison: Evaluation allows for the comparison of different algorithms or model versions to select the best-performing one

Recommenders Accuracy Metrics: RMSE

	p_1	p_2	p_3						p_j									p_m
u_1	1		2	1		4			2		3	2		5				2
u_2		1				2		1			2			1				3
u_3		1	1	2				1							1		2	
				3		2			5			2			3	4		
		1			2									5				
u_i			3	2	1		4	5		1		3	1		2			1
			4													4		
			5			1										5		
		1		4							1	3		5		1	2	
u_n			3		1	1		2	1					4				5

Test Set

Compare predictions $U'(i,j)$ with known (hidden) ratings

Root-mean-squared-error, RMSE $= \sqrt{\sum_{i,j \in \text{Test Set}} (U(i,j) - U'(i,j))^2 / |\text{Test Set}|}$

▷ RMSE is sensitive to outliers and does not account for ranking

▷ RMSE does not capture user satisfaction

Recommenders Accuracy Metrics: Precision, Recall, and F1 Score

- Precision:

$$\text{Precision} = \frac{\text{Number of Relevant Items Recommended}}{\text{Total Number of Recommended Items}}$$

- Recall:

$$\text{Recall} = \frac{\text{Number of Relevant Items Recommended}}{\text{Total Number of Relevant Items}}$$

- F1 Score:

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Spearman's rank correlation coefficient

A measure of similarity between two variables based on rank

- **Correlation between ranks of values of the variables**

$$\rho_{xy} = \frac{\mathrm{COV}(rg_X, rg_Y)}{\sigma_{rg_X}\sigma_{rg_Y}}$$

X	3	3	3	4	4	4	5	5	5	0	0	1	1	2	2	2
Y	4	3	4	5	4	5	4	4	3	0	1	0	1	3	2	3
Z	1	0	1	2	1	2	1	1	0	3	4	3	4	0	5	0

Spearman's rank correlation coefficient

A measure of similarity between two variables based on rank

- **Correlation between ranks of values of the variables**

$$\rho_{XY} = \frac{\mathrm{COV}(rg_X, rg_Y)}{\sigma_{rg_X}\sigma_{rg_Y}}$$

X	3	3	3	4	4	4	5	5	5	0	0	1	1	2	2	2
Y	4	3	4	5	4	5	4	4	3	0	1	0	1	3	2	3
Z	1	0	1	2	1	2	1	1	0	3	4	3	4	0	5	0
rg_X	8	9	10	11	12	13	14	15	16	1	2	3	4	5	6	7
rg_Y	10	6	11	15	12	16	13	14	7	1	3	2	4	8	5	9
rg_Z	4	5	6	7	8	9	10	11	12	13	15	14	16	1	2	3

$$\rho_{XY} = 0.8 \qquad \rho_{XZ} = -0.1 \qquad \rho_{YZ} = -0.3$$

Diversity Metrics

Diversity metrics measure how varied the recommendations are, ensuring that users are not only recommended similar items.

High diversity in recommendations can lead to a more engaging user experience - prevents the recommendater from becoming too narrow or repetitive:

- Intra-list Diversity: Measures how dissimilar the items in a recommendation list are to each other
- Category Coverage/Spread: Measures how many different categories or genres are represented in the recommendations
- Entropy: Quantifies the uncertainty or randomness in the recommendations. Higher entropy indicates greater diversity

Novelty Metrics

Novelty metrics measure a recommender's ability to suggest items that are new or less obvious to the user, but still relevant..

Novelty ensures the user is exposed to a broader range of items:

Novelty encourages exploration and helps users discover new, less mainstream, highly relevant, and interesting content

Can surprise users with suggestions outside their typical interests, leading to a richer discovery experience

- Long Tail Recommendations: Recommending less popular items, ensures a broader catalog is explored
- Novelty at k: Measures how often the top-k recommendations include items the user has not interacted with before.

Serendipity Metrics

Serendipity is a measure of how pleasantly surprised users are by the recommendations

While similar to novelty, serendipity focuses on unexpected but delightful recommendations

Achieving serendipity improves user engagement by offering both relevance and surprise.

Encourages the model to explore less obvious items

Mixes popular items with less common but highly personalized recommendations

- Serendipity Score: Measures how often recommendations are both unexpected and relevant
- User Feedback: Direct user feedback can be used to assess whether recommendations are pleasantly surprising

Experimental Design

Designing experiments for evaluating the effectiveness of recommenders

- Offline Evaluation: Testing models using historical data
  - Train-Test Split
  - Cross-Validation
- Online Evaluation: Testing models using live user interactions
  - A/B Testing
  - Online Metrics: Measure click-through rates, conversion rates, and user engagement in real-time
- User Studies: Gather qualitative users' feedback
  - Surveys and Interviews: Collect user feedback on the relevance, diversity, and novelty of recommendations
  - Focus Groups: Bring together users to discuss their experiences and provide detailed feedback

Recommendation System Challenges

Recommendation Systems: Challenges

- **Gather** "known" ratings (populate matrix U)
- **Extrapolate** unknown ratings from known ones

  $\triangleright$ mainly interested in high ratings, Top k

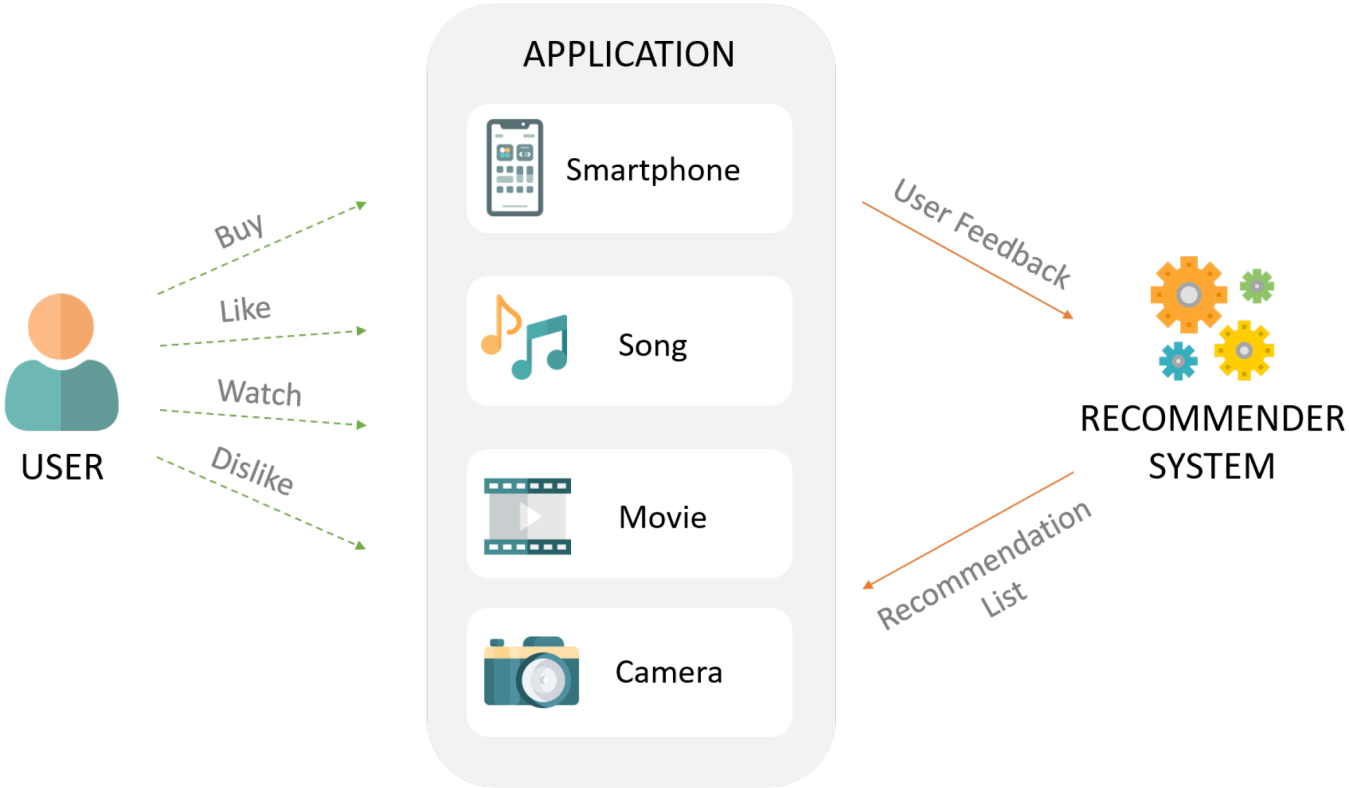
  - For each user c or a subset of users, find

$$R_c = \arg\max_p U(c, p)$$

  - R_c is the recommendation(s) for user c
- **Evaluate** extrapolation methods

Recommendation Systems: Gathering Data

- Explicitly survey users (e.g., movie ratings or product reviews)
- Implicitly learn ratings, e.g. clicks, views, purchases/suggestion to friend/watch time implies high rating
- Cold-start problem (new user, new product)



Challenges in Recommendation: Gathering Data

Effective recommendation systems require rich, diverse data sources.

Gathering data is challenging due to:

- Data Sparsity: Users only interact with a small subset of items, leading to sparse user-item interaction matrices
- Noisy Data: Implicit feedback data, such as clicks or views, might not always represent true preferences
- Incomplete Data: Missing data, such as unrecorded ratings or interactions, can make it difficult to provide accurate predictions

Challenges in Recommendation: Cold Start Problem

The cold start problem arises when the system has little or no data about a new user or item

- New Users: Without sufficient interaction data, it is challenging to recommend items
- New Items: It is difficult to recommend new items as the system lacks information about user preferences for these items

Challenges in Recommendation: Extrapolation

Extrapolation: Predicting preferences for items or users for which limited data exists.

It can be particularly challenging when:

- Data is Sparse: When a user or item has few interactions, it becomes difficult to generalize preferences
- Item Popularity: Extrapolation often favors popular items, leading to a lack of diversity in recommendations

Challenges in Recommendation: Biases in Model Predictions

Bias in recommendation systems occurs when the model's predictions are skewed, leading to inaccurate or "unfair" recommendations

Common sources of bias:

- Popularity Bias: Recommending popular items over niche ones, reducing recommendation diversity
- Interaction Bias: Implicit feedback may overrepresent frequently interacted items, leading to biased recommendations
- Algorithmic Bias: The system may favor certain users or groups based on the underlying training data

Fairness-aware algorithms and diversity metrics "mitigate" these biases

Recommendation using Averages and ANOVA

Raw Averages Based Recommendation

Raw averages-based recommendation is a simple interpretable approach to generating predictions

- Global Average: Assign average rating of all users for any item (GLOBAL-AVERAGE)
- User Average: Assign average rating of all items by user u, (USER-AVERAGE)
- Item Average: Assign average rating of all users for item j, (ITEM-AVERAGE)

Issues with averages based matrix completion

1. They often overlook individual user preferences and interactions between users and items
2. Mean is an unstable statistics (could use other measures of location)

Recommendation Methods: Global Average

Predicting $U(i,j)$

Let MoM be the matrix mean (mean of means/ global mean)

$$\text{MoM} := \frac{1}{|\text{Train Set}|;} \sum_{c,p \in \text{ Train Set}} U(c,p), \quad \text{then}$$

$$U'(i,j) = \text{MoM}$$

RMSE is just the standard-deviation of the data (train set)

The Complete Rating Matrix

	p_0	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}
u_0	2	3	3	4	2	3	4	5	5	4	3	2	4
u_1	3	3	2	4	4	3	4	5	5	4	4	3	5
u_2	2	3	3	4	3	4	5	5	4	3	2	4	5
u_3	3	4	4	5	4	5	5	5	5	5	4	3	5
u_4	2	3	3	4	4	4	5	5	5	5	4	3	4
u_5	3	4	4	5	4	5	5	5	5	4	4	3	5
u_6	3	4	4	5	5	5	5	5	5	5	4	4	5
u_7	4	5	5	5	5	5	5	5	5	5	5	4	5
u_8	3	4	4	5	5	5	5	5	5	5	4	4	5
u_9	4	5	5	5	5	5	5	5	5	5	5	4	5

Recommendation Methods: Averages and ANOVA

	p_0	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}
u_0	2	3	3	4	2	3	4	5	5	4	3	2	4
u_1	3	3	2	4	4	3	4	5	5	4	4	3	5
u_2	2	3	3	4	3	4	5	5	4	3	2	4	5
u_3	3	4	4	5	4	5	5	5	5	5	4	3	5
u_4	2	3	3	4	4	4	5	5	5	5	4	3	4
u_5	3	4	4	5	4	5	5	5	5	4	4	3	5
u_6	3	4	4	5	5	5	5	5	5	5	4	4	5
u_7	4	5	5	5	5	5	5	5	5	5	5	4	5
u_8	3	4	4	5	5	5	5	5	5	5	4	4	5
u_9	4	5	5	5	5	5	5	5	5	5	5	4	5

Users' Tendencies:

- u_0: Highly Pessimistic
- u_1: Moderately Pessimistic
- u_2: Slightly Pessimistic
- u_3: Neutral
- u_4: Slightly Optimistic
- u_5: Moderately Optimistic
- u_6: Highly Optimistic
- u_7: Always Positive
- u_8: Optimistic
- u_9: Extremely Positive

Products' Qualities:

- p_0: Poor
- p_1 and p_2: Below Average
- p_3 and p_4: Average
- p_5 and p_6: Above Average
- p_7 and p_8: Good
- p_9 and p_{10}: Very Good
- p_{11} and p_{12}: Excellent

Masked Rating Matrix with 30% Missing Values for Validation

	p_0	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}
u_0	2	3	?	4	2	?	4	5	5	?	3	2	4
u_1	3	3	2	?	4	3	?	5	5	4	?	3	5
u_2	2	?	3	4	?	4	5	5	?	3	2	4	5
u_3	3	4	4	5	4	?	5	5	5	5	4	3	5
u_4	2	3	3	4	?	4	?	5	5	5	4	?	4
u_5	?	4	4	5	4	?	5	5	5	4	4	3	5
u_6	3	4	?	5	5	5	5	5	5	?	4	4	5
u_7	4	5	5	?	5	5	?	5	5	5	5	4	5
u_8	3	?	4	5	5	5	5	?	5	5	4	4	5
u_9	4	5	?	5	5	5	5	5	?	5	5	4	5

Recommendation Methods: Global Average

Missing entries filled with **Global Average:** $MoM = 4.233$

	p_0	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}
u_0	2	3	4.23	4	2	4.23	4	4.23	5	4.23	3	2	4
u_1	3	3	2	4.23	4	3	4.23	5	5	4	4.23	3	5
u_2	2	4.23	3	4	4.23	4	5	5	4.23	3	2	4	5
u_3	3	4	4	5	4	4.23	5	5	5	5	4	3	5
u_4	2	3	3	4	4.23	4	4.23	5	5	5	4	4.23	4
u_5	4.23	4	4	5	4	4.23	5	5	5	4	4	3	5
u_6	3	4	4.23	5	5	5	5	5	5	4.23	4	4	5
u_7	4	5	5	4.23	5	5	4.23	5	5	5	5	4	5
u_8	3	4.23	4	5	5	5	5	4.23	5	5	4	4	5
u_9	4	5	4.23	5	5	5	5	5	4.23	5	5	4	5

RMSE: 1.195166

Recommendation Methods: User Average

Missing entries filled with User Averages (Row-wise Averages)

	p_0	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}
u_0	2	3	3.4	4	2	3.4	4	3.4	5	3.4	3	2	4
u_1	3	3	2	3.7	4	3	3.7	5	5	4	3.7	3	5
u_2	2	3.7	3	4	3.7	4	5	5	3.7	3	2	4	5
u_3	3	4	4	5	4	4.33	5	5	5	5	4	3	5
u_4	2	3	3	4	4.3.9	4	3.9	5	5	5	4	3.9	4
u_5	4.36	4	4	5	4	4.36	5	5	5	4	4	3	5
u_6	3	4	4.54	5	5	5	5	5	5	4.54	4	4	5
u_7	4	5	5	4.82	5	5	4.82	5	5	5	5	4	5
u_8	3	4.54	4	5	5	5	5	4.54	5	5	4	4	5
u_9	4	5	4.82	5	5	5	5	5	4.82	5	5	4	5

RMSE: 1.087284456

Recommendation Methods: Item Average

Missing entries filled with Item Averages (Column-wise Averages)

	p_0	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}
u_0	2	3	3.57	4	2	4.43	4	5	5	4.5	3	2	4
u_1	3	3	2	4.62	4	3	4.86	5	5	4	3.89	3	5
u_2	2	3.87	3	4	4.25	4	5	5	5	3	2	4	5
u_3	3	4	4	5	4	4.42	5	5	5	5	4	3	5
u_4	2	3	3	4	4.25	4	4.86	5	5	5	4	3.44	4
u_5	2.9	4	4	5	4	4.43	5	5	5	4	4	3	5
u_6	3	4	3.57	5	5	5	5	5	5	4.5	4	4	5
u_7	4	5	5	4.62	5	5	4.86	5	5	5	5	4	5
u_8	3	3.87	4	5	5	5	5	5	5	5	4	4	5
u_9	4	5	3.57	5	5	5	5	5	5	5	5	4	5

RMSE: 1.157043487

Measures of Central Tendencies: Mean

For a dataset $\mathcal{X} = \{x_1, x_2, \cdots, x_n\}$

- **(Arithmetic) Mean** is the average of the data set
  - ▷ This definition readily extend to higher dimensional data

$$\overline{x} = \frac{x_1 + x_2 + \ldots + x_n}{n} = \frac{\sum_{i=1}^{n} x_i}{n}$$

- **Harmonic Mean**

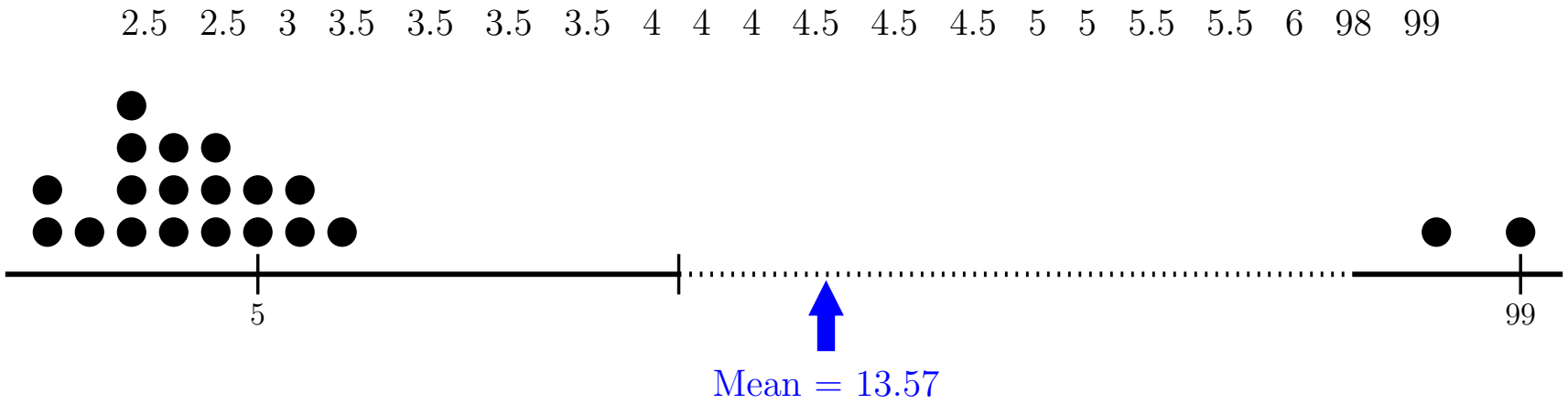
$$\overline{x} = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$$

- **Geometric Mean**

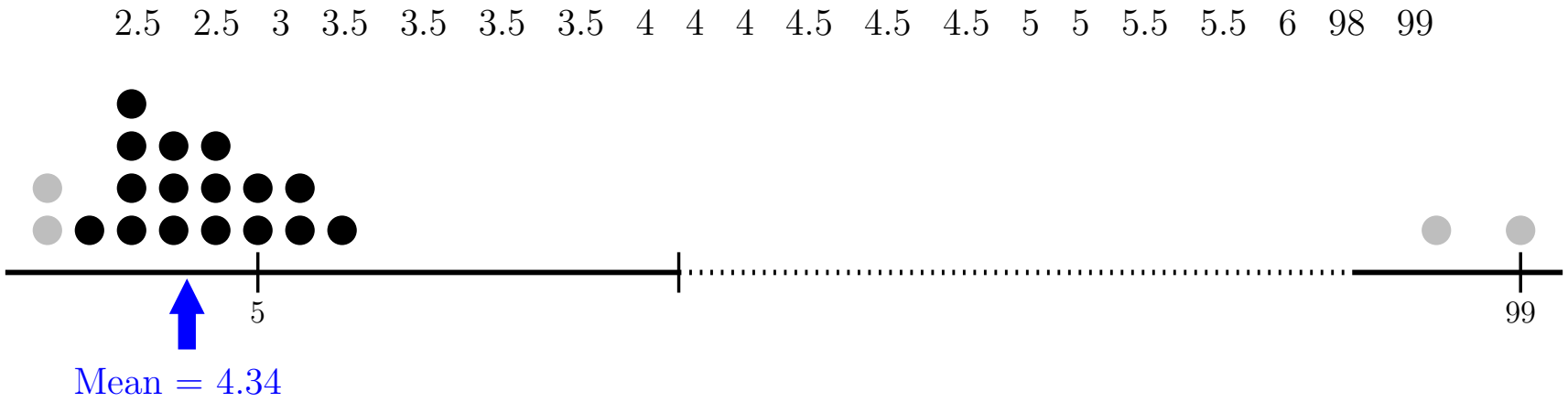
$$\overline{x} = \left(\prod_{i=1}^{n} x_i \right)^{1/n}$$

Measures of Central Tendencies:Trimmed or Truncated Mean

- Arithmetic mean is sensitive to outliers ▷ unstable statistic
- Just one very high/low value (think $\pm\infty$) makes mean very high/low



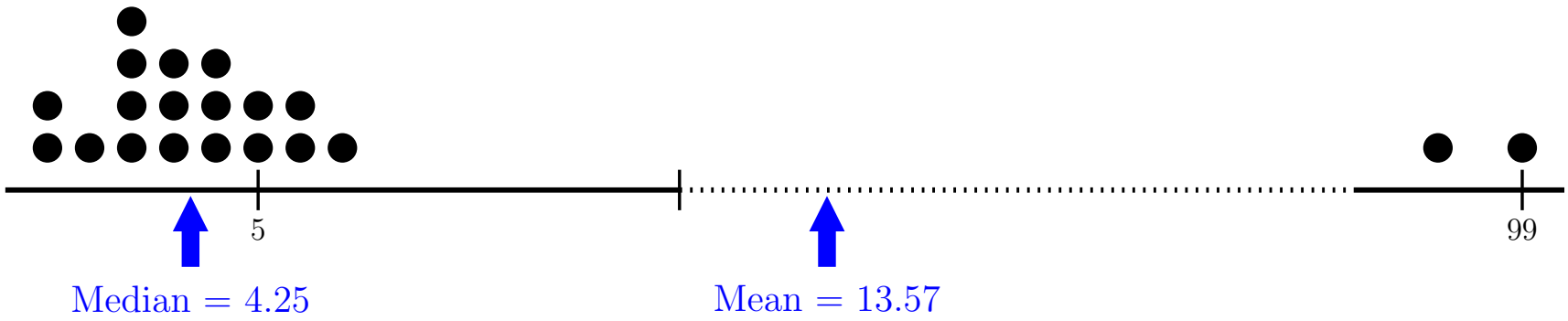
Trimmed Mean: Ignore $k\%$ of values at both extremes to compute mean



Measures of Central Tendencies: Median

Median: The value that divides the ratings into two equal halves

- Median is less sensitive to outliers as compared to mean
- Median is good for asymmetric distributions and where data has outliers



- Various possible definitions for median of higher dimensional data
- Mean together with variance (see below) has nice properties

Recommendation Methods: ANOVA

More sophisticated method ANOVA improve upon simple raw averages

ANOVA (Analysis of Variance): A statistical method used to determine the influence of one or more independent variables on a dependent variable

Variance analysis in recommenders is used to understand how different factors contribute to variations in ratings. We might observe:

- User variance: Some users are more lenient or harsher in their ratings
- Item variance: Some items may consistently receive high or low ratings across users
- Interaction variance: Variability in ratings based on the specific user-item interaction

ANOVA decomposes total variance into how much is due to users/items

Improves prediction by focusing on factors contributing to rating variability

Recommendation Methods: ANOVA

Predicting $U(i, j)$

- Idea: Assign global average (matrix mean) $U(i, j) = \text{MoM}$
- Refinement 1: Product j maybe very (un) popular – highly (un)liked
  - Adjust for this bias
- Let dev_j be the average deviation of item j from MoM (+ve or -ve)

$$U(i, j) = \text{MoM} + dev_j$$

- Refinement 2: User i may be very (non) pessimistic (critical)
  - Adjust for this bias too
- Let dev_i be the average deviation of user i from MoM

$$U(i, j) = \text{MoM} + dev_j + dev_i$$

Other methods are generally compared with this baseline

Recommendation Methods: Basic ANOVA

Global Mean (MoM) = 4.233

User Deviations from MoM:

- u_0: -0.83
- u_1: -0.53
- u_2: -0.53
- u_3: -0.099
- u_4: -0.33
- u_5: 0.13
- u_6: 0.31
- u_7: 0.58
- u_8: 0.31
- u_9: 0.58

Item Deviations from MoM:

- p_0: -0.13
- p_1: -0.35
- p_2: -0.66
- p_3: 0.39
- p_4: 0.016
- p_5: 0.194
- p_6: 0.623
- p_7: 0.766
- p_8: 0.766
- p_9: 0.266
- p_{10}: -0.344
- p_{11}: -0.789
- p_{12}: 0.566

Recommendation Methods: Basic ANOVA

Missing entries willed with ANOVA (Adjusted for user and item deviations)

	p_0	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}
u_0	2	3	2.73	4	2	3.6	4	5	5	3.66	3	2	4
u_1	3	3	2	4.09	4	3	4.32	5	5	4	3.35	3	5
u_2	2	3.34	3	4	3.72	4	5	5	4.47	3	2	4	5
u_3	3	4	4	5	4	4.53	5	5	5	5	4	3	5
u_4	2	3	3	4	3.92	4	4.52	5	5	5	4	3.11	4
u_5	3.01	4	4	5	4	4.56	5	5	5	4	4	3	5
u_6	3	4	3.88	5	5	5	5	5	5	4.81	4	4	5
u_7	4	5	5	4.65	5	5	4.78	5	5	5	5	4	5
u_8	3	4.18	4	5	5	5	5	**5.0**	5	5	4	4	5
u_9	4	5	4.15	5	5	5	5	5	**5.0**	5	5	4	5

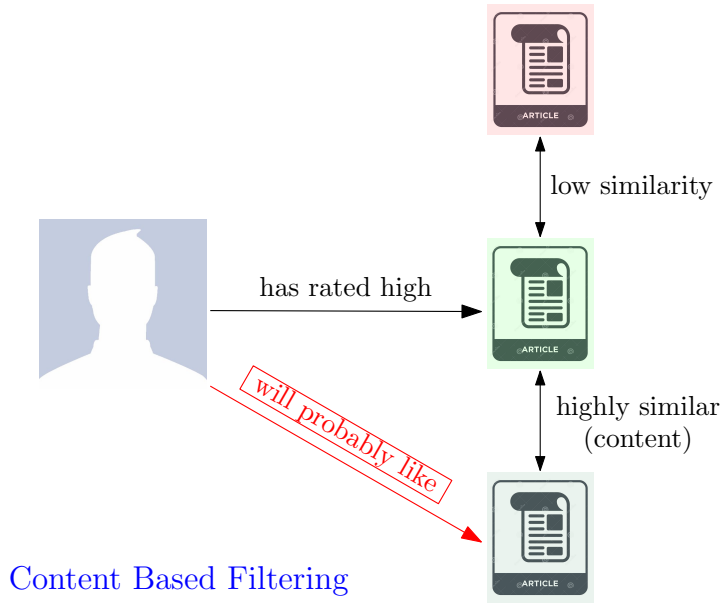
RMSE: 1.04

Content Based Filtering

Recommendation Methods: Content-based

Content Based Filtering: Recommend items based on the characteristics of the items themselves and the preferences or profile of the user

Utilizes item features and user interaction history to predict items that are similar to what the user has liked in the past



No need for large user interaction data and can provide personalized recommendations

Predict $U(i, j)$ high, if j is similar to the "taste" of i

1. Build Item Profile (based on content) e.g.
   - movies: vector of genre, director, budget, cast, plot, language
   - books, blogs, website, news items: TF-IDF vector, author, topic
2. Build User Profile
   - **A vector with the same coordinates as item profile**
   - kind of "an average item" that the user likes ▷ the taste of user
   - Weighted (by ratings) average of the item profiles that the user has rated
3. $U'(i, j) \propto$ (cosine) similarity between item j's and user i's profiles

Recommendation Methods: Content-based

Predict $U(i, j)$ high, if j is similar to the "taste" of i

Pros

- No need of other users' information
- No cold-start or sparsity problem w.r.t items
- Unique taste of user is captured, personalized recommendations
- Able to provide explanation (by listing contents' features)

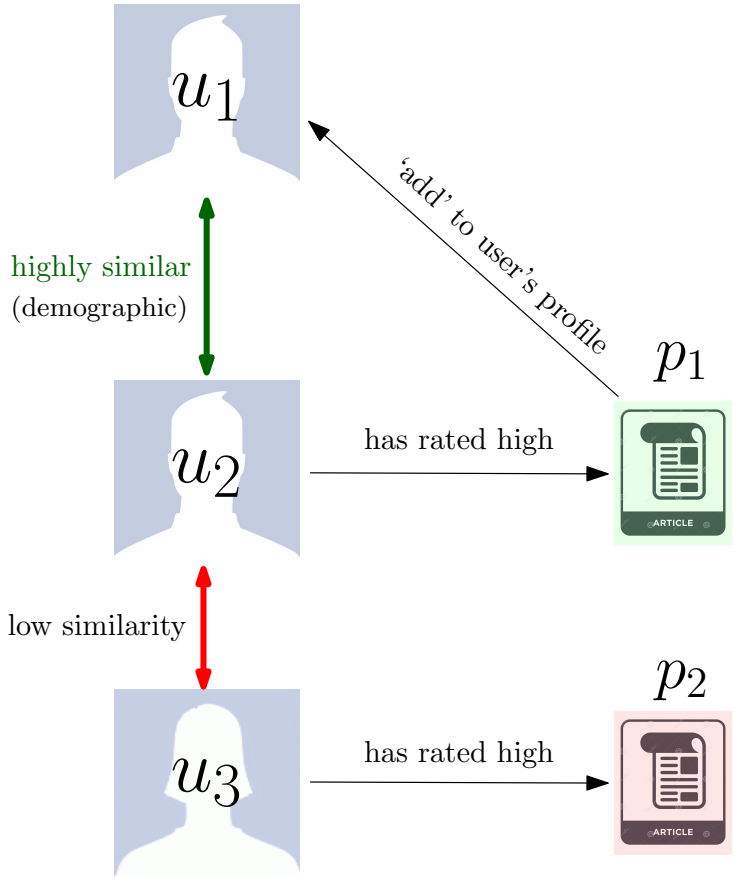
Cons

- Building profile is hard, finding relevant features is hard
- Cold start problem w.r.t users
- User profile is heuristic
- Overspecialization-never recommends items outside user profile
- Does not cater for multiple interests of a user
- Does not utilize judgment of other users

Predict $U(i, j)$ high, if j is similar to the "taste" of i

Can take into account other (similar) users judgments as follows

▷ Somewhat cater for the cold start problem



Item Profiles

Item profiles: Feature vector of item in the system

- Each item is described by a set of features (e.g., genre, author, price).
- Compare items profile to recommend similar ones to the user

Feature extraction is a key step in building item profiles:

- Textual Data: For movies or books, keywords, genre, or descriptions are extracted using techniques like TF-IDF or word embeddings
- Multimedia Content: For images or videos, features include resolution, format, or visual features extracted through computer vision techniques
- Categorical Data: For products, categorical features like brand, price range, or category are used to build item profiles

User Profiles

User profiles: the preferences of user, built from historical ratings

- Generated by analyzing the profiles of items the user has rated
- A weighted feature vector summarizing the user's preferences (e.g., a user has a high preference for the "comedy" genre in movies)

User preference modeling in content-based filtering involves:

- Aggregating Features: The system aggregates features from items that the user has rated (e.g., the genres of movies watched).
- Creating a Preference Vector: A weighted vector is formed based on the user's past interactions, assigning higher weights to the most frequently rated features.
- Dynamic Updating: The user profile updates as new interactions occur, adjusting the user's preferences over time.

Similarity Measures

To recommend items, content-based filtering systems calculate the similarity between items or between a user and items:

- Cosine Similarity: Measures the cosine of the angle between two vectors, often used for textual or categorical data.
- Euclidean Distance: Measures the straight-line distance between two vectors in a feature space, useful for numerical features.

The similarity score is then used to rank items for recommendation.

In a document recommender, cosine similarity can be used to compare the keywords of two articles to determine if they are similar.

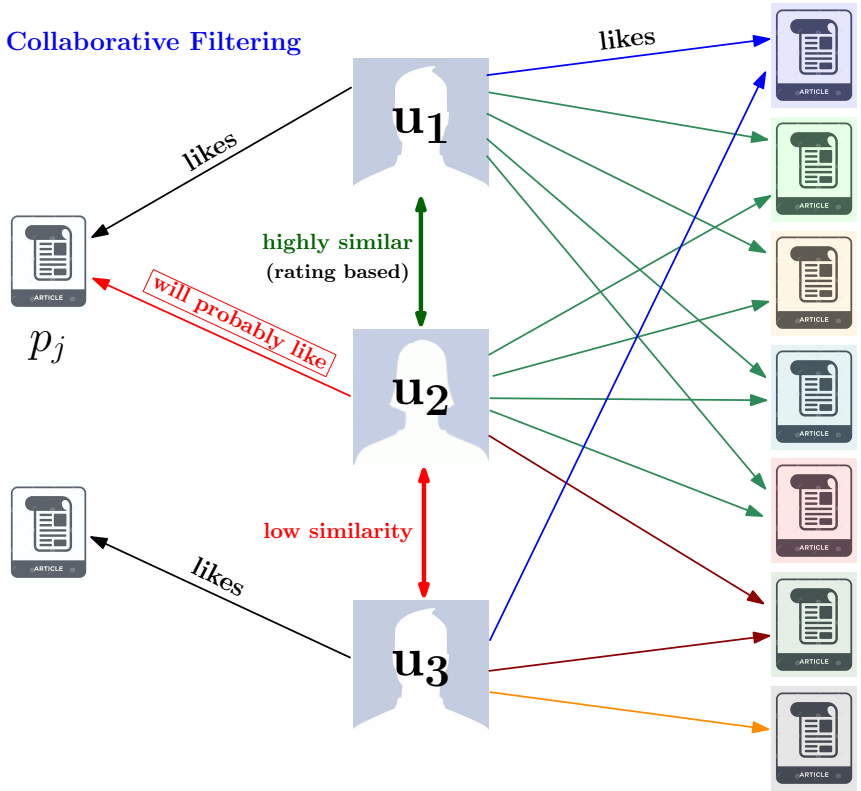
In a product recommender, Euclidean distance can be used to compare feature vectors like price, brand, and category.

Neighborhood Model: Collaborative Filtering

Collaborative Filtering: Neighborhood Model

Collaborative filtering a widely used method for recommendation, leverages the collective wisdom of a group of users or items to make predictions

Predicts user preferences by analyzing the preferences of similar users (user-user) or similar items (item-item)



User-User Collaborative Filtering

Recommends items to a user based on the preferences of "similar" users

▷ Assumption: Users with similar preferences in the past will continue to have similar preferences in the future

- Find the set N of users with similar ratings as of i
  - Find the top k similar rows to the ith row
- Estimate $U(i,j)$ as an "average" of $U(a,j)$'s for $a \in N$
- i has similar 'taste' to $a \in N \implies U(i,j)$ similar to $U(a,j)$

  Predict $U(i,j)$ based on weighted average of ratings of similar users

$$U'(i,j) = \frac{\sum_{a \in N} sim(a,i) \times U(a,j)}{\sum_{a \in N} sim(a,i)}$$

OR $$U'(i,j) = \bar{r}_i + \frac{\sum_{a \in N} \text{sim}(a,i)(r_{ai} - \bar{r}_a)}{\sum_{a \in N} |\text{sim}(a,i)|}$$

User-User Collaborative Filtering: Similarity Measures

User-user CF use similarity measures to find users with similar preferences

- Cosine similarity:
- Pearson Correlation: Measures linear correlation in users' rating

$$\text{sim}_{\rho}(u, v) = \frac{\sum_{i \in I}(r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I}(r_{ui} - \bar{r}_u)^2}\sqrt{\sum_{i \in I}(r_{vi} - \bar{r}_v)^2}}$$

  - r_{ui} and r_{vi} are the ratings of users u and v for item i.
  - $\bar{r}_u$ and $\bar{r}_v$ are the average ratings of users u and v.
  - I is the set of items rated by both users.
- Jaccard Index: Measures similarity between sets (clicks or views)

$$\text{sim}_{Jaccard}(u, v) = \frac{|I_u \cap I_v|}{|I_u \cup I_v|}$$

  - I_u and I_v are sets of items interacted with by users u and v, respectively

Item-Item Collaborative Filtering

Item-item CF recommends items by analyzing similarities between items

Predicts how much a user will like an item based on how similar it is to other items the user has rated or interacted with

- Find the set W of items similarly rated as j
  - Find the top k similar columns to the jth row
- Estimate $U(i,j)$ as an "average" of $U(i,p)$'s for $p \in W$

$$U'(i,j) = \frac{\sum_{p \in W} U(i,p) \times sim(j,p)}{\sum_{p \in W} sim(j,p)}$$

- Better result by item-item collaborative filtering
  - Because items are easier to model
  - has less complexity than users
  - item-item relationships are more stable over time than user preferences

Item-Item Collaborative Filtering: Similarity Measures

Item-item CF use similarity measures between items

- Cosine Similarity: Cosine similarity is widely used in recommender systems to measure the similarity between users or items based on their feature representations. It quantifies how similar two vectors (e.g., user preference vectors) are by computing the cosine of the angle between them.
- Pearson Correlation: Linear correlation between the ratings of two items across users
- Co-occurrences: Measures how often two items are interacted with by the same users, very useful for implicit feedback scenarios
  - Co-occurrence Matrix: Constructed by counting how often two items are interacted with by the same users

    ▷ two items that are frequently bought together by the same users (like a camera and a memory card) will have a high co-occurrence score
  - Jaccard Index: Measures overlap in users who interacted with two items

Implicit Feedback in CF

Incorporating implicit feedback is usefule, esp. when explicit ratings are unavailable or sparse

- Clicks: Items the user clicked on.
- Views: Items the user viewed.
- Purchases: Items the user bought.

This data is often noisier but abundant, allowing for richer insights into user behavior

Recommenders systems treats these interactions as indicators of interest

- Weighted Implicit Feedback: Clicks or views weighted less heavily than actual purchases or ratings
- Interaction Frequency: Frequently viewed or clicked item receive a higher relevance score

Advantages and Disadvantages of Collaborative filtering

- No Domain Knowledge Required: CF doesn't rely on item features, making it flexible for applications with any kind of items
- Diversity: The system can make recommendations outside a user's usual preferences by leveraging group wisdom
- Serendipity: CF can introduce unexpected but relevant items, encouraging users to explore content outside their usual preferences

- Cold-Start problem: Struggles to recommend for new users or items without interaction history.
- Sparsity: Most users interact with a small fraction of items, hard to find users that have rated the same items
- First rater: cannot recommend items that are not previously rated (new or esoteric items)
- Scalability Issues: Finding similar users/items is computationally expensive for large datasets

Approximate nearest neighbors (e.g., LSH) and dimensionality reduction (e.g., SVD) help improve scalability in CF

Recommendation using Matrix Factorization

Matrix Factorization Based Recommendation Systems

Recall the recommendation system problem is the problem of matrix completion in computational linear algebra

Given a rating matrix R – users ratings for items, predict $R(i, j)$

	p_1	p_2	p_3						p_j									p_m
u_1	1		2	1		4			2		3	2		5				2
u_2		1				2		1			2			1				3
u_3		1	1	2				1							1		2	
				3		2			5			2			3	4		
		1			2									5				
u_i			3	2	1		4	5	?	1		3	1		2			1
			4													4		
			5			1										5		
		1		4							1	3		5		1	2	
u_n			3		1	1		2	1					4				5

Matrix factorization based recommenders decompose the user-item interaction matrix into latent factors, to predict its missing values

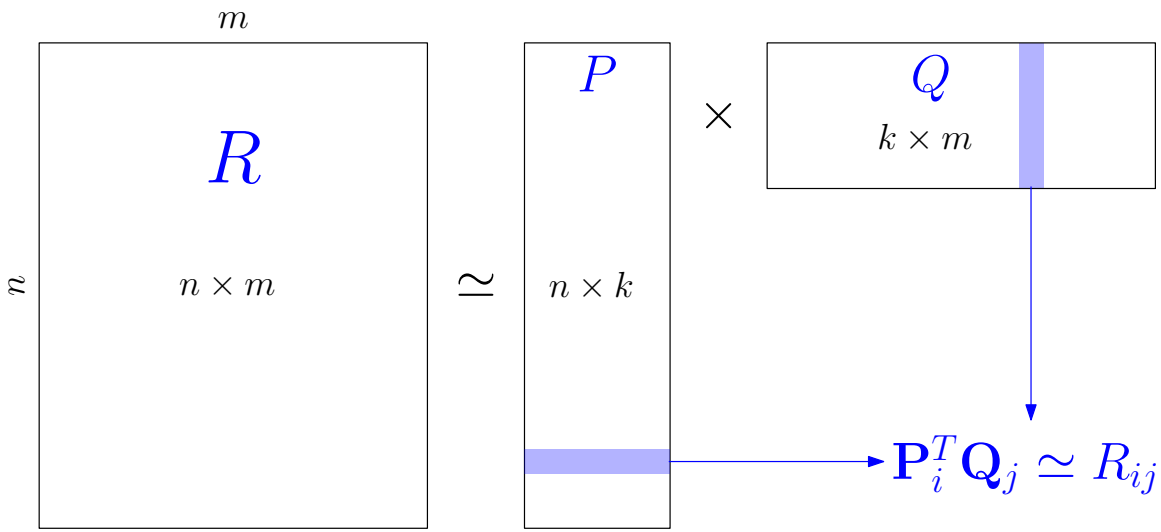
Matrix Factorization Based Recommendation Systems

- Given $n \times m$ matrix R For $k \ll m, n$, Find
- $n \times k$ matrix P and $k \times m$ matrix Q such that

$$R = PQ$$

Generally, for very small k, we seek

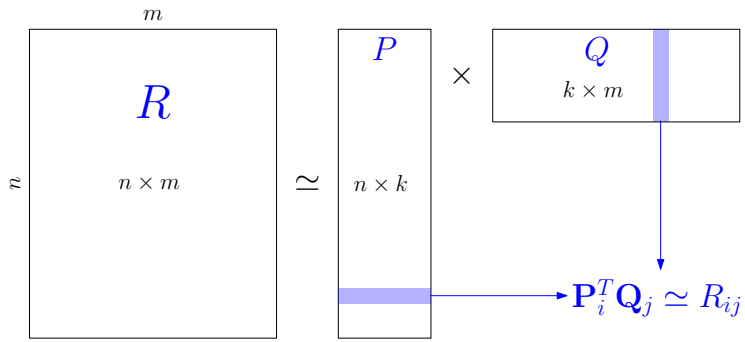
$$R \simeq PQ$$



Matrix Factorization Based Recommendation Systems

- Given $n \times m$ matrix R For $k \ll m, n$, Find
- $n \times k$ matrix P and $k \times m$ matrix Q such that

$$R \simeq PQ$$



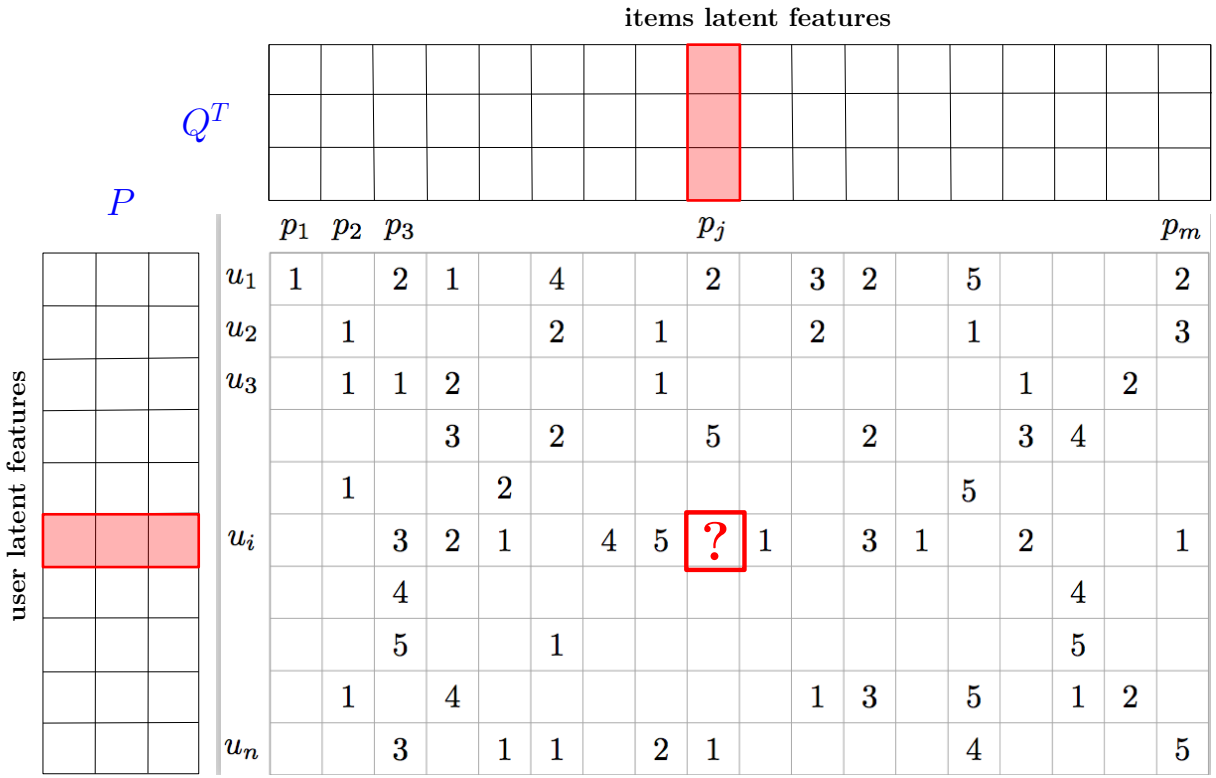
This is a classic optimization problem can be solved as

$$\min_{\substack{P \in \mathbb{R}^{n \times k} \\ Q \in \mathbb{R}^{m \times k}}} \sum_{(i,j)} \left(R_{ij} - P_i Q_j^T \right)^2 + \underbrace{\lambda \left(\|P\|_F^2 + \|Q\|_F^2 \right)}_{\substack{\text{regularization term} \\ \text{avoids overfitting}}}$$

Later we will discuss low rank approximation (SVD) to solve this problem

Matrix Factorization for Recommenders $R \simeq PQ$

- P : k-dim representation of users in a latent feature space $\mathbb{R}^k$
- Q : k-dim representation of items latent feature space
- $P_i Q_j^T$: interaction between user i and item j – approximation of R_{ij}



Matrix Factorization Based Recommendation Systems

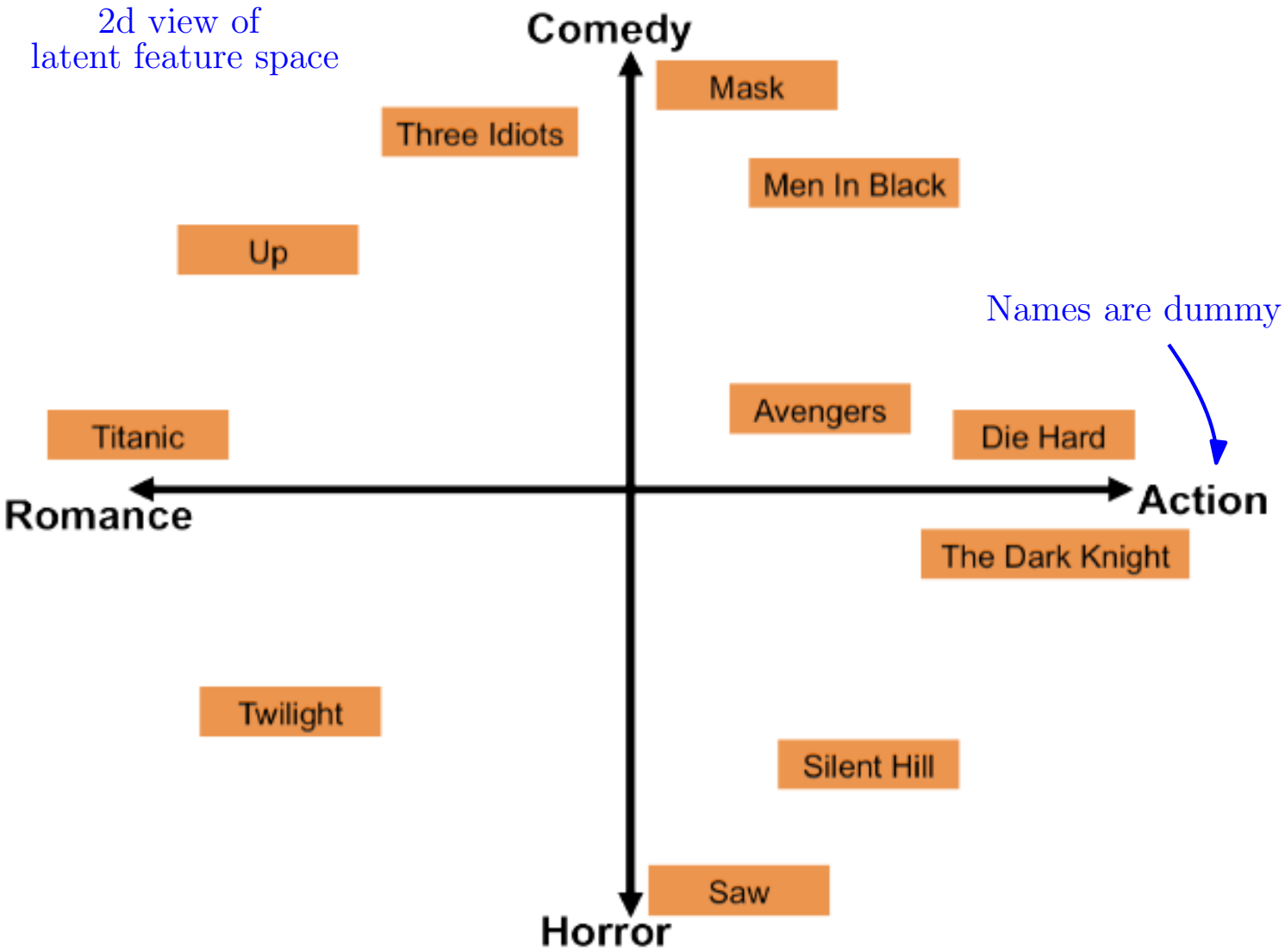


diagram adapted from Cho-Jui Hsieh @ UCLA

Matrix Factorization Based Recommendation Systems

Users and movies mapped to latent feature space

Points are coordinates of users and movies

(2-dim P and Q)

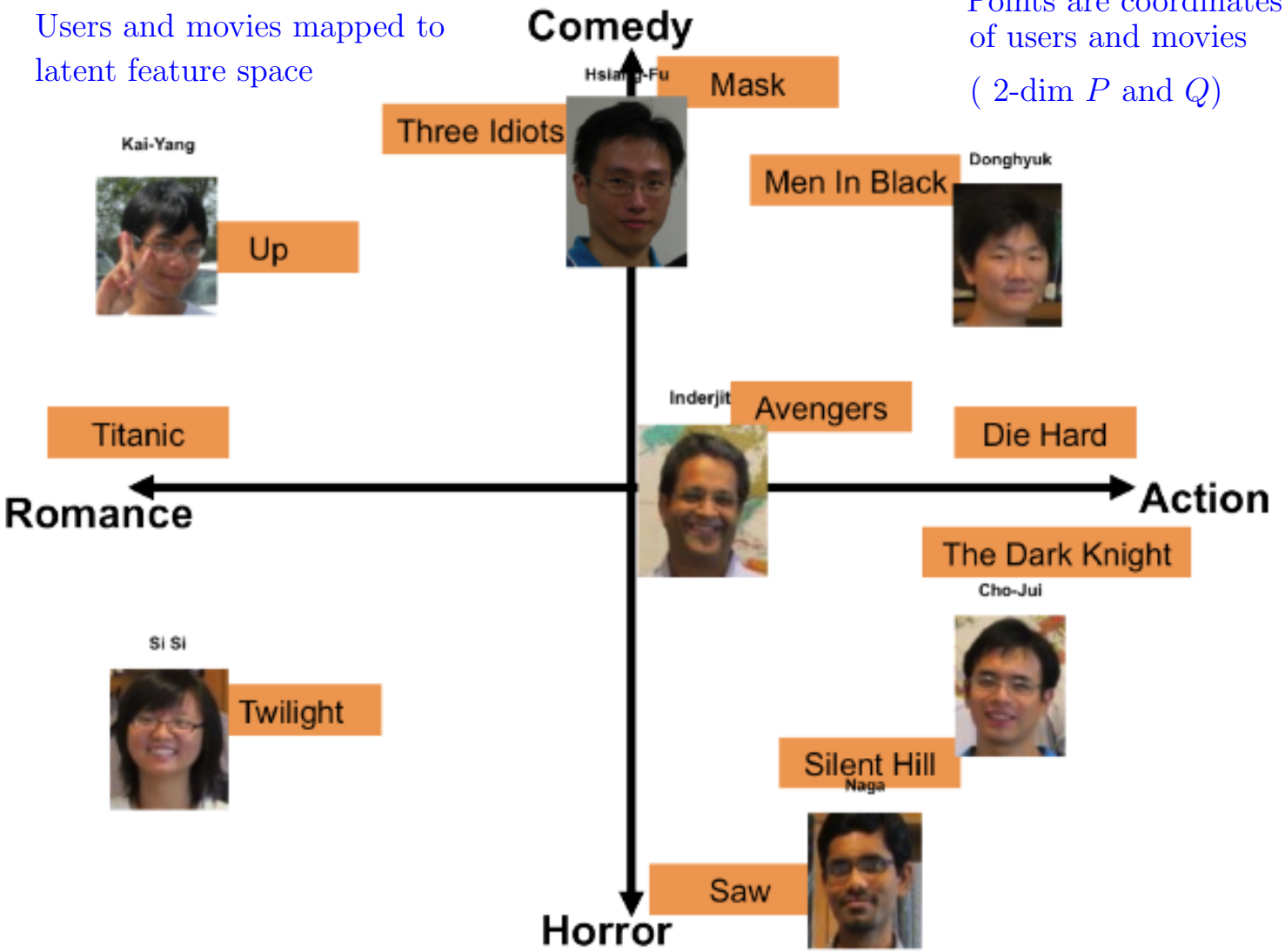


diagram adapted from Cho-Jui Hsieh @ UCLA

Rank Factorization of Matrices

Rank of a matrix

For an $n \times m$ matrix A

Column Rank of A, col-rank(A) is the maximum number of linearly independent columns of A

Row Rank of A, row-rank(A) is the maximum number of linearly independent rows of A

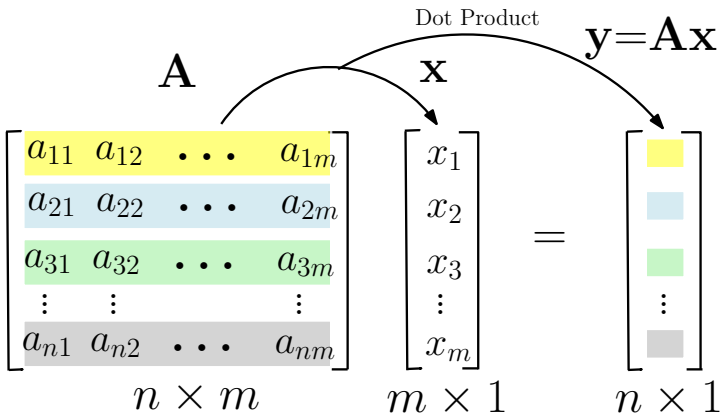
$$rank(A) := \text{col-rank}(A) = \text{row-rank}(A)$$

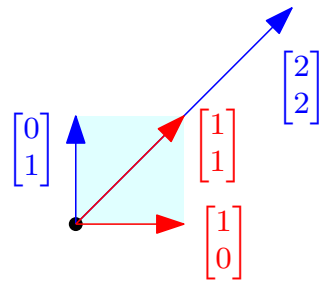
Rank of a matrix

For an $n \times m$ matrix A

Looking at A as a linear transformation i.e. $A : \mathbb{R}^m \mapsto \mathbb{R}^n$

$rank(A)$ is the true dimensionality of the range (output) space of A





columns of $\begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix}$ are linearly dependent

If $rank(A) = k$, then output vectors live in a k-d subspace

Rank of a matrix

Another definition of rank (aka decomposition rank)

An $n \times m$ matrix A has

Rank-0 if all its entries are 0

Rank-1 if it is outer product of an $n \times 1$ and an $m \times 1$ vector, $A = \mathbf{u}\mathbf{v}^T$

$$A = \mathbf{u}\mathbf{v}^T = \begin{bmatrix} | \\ \mathbf{u} \\ | \end{bmatrix} \begin{bmatrix} — & \mathbf{v}^T & — \end{bmatrix} = \begin{bmatrix} | & | & \dots & | \\ v_1\mathbf{u} & v_2\mathbf{u} & \dots & v_m\mathbf{u} \\ | & | & \dots & | \end{bmatrix} = \begin{bmatrix} — & u_1\mathbf{v}^T & — \\ — & u_2\mathbf{v}^T & — \\ \vdots & \ddots & \vdots \\ — & u_n\mathbf{v}^T & — \end{bmatrix}$$

Rank-2 if it is non-trivial sum of two rank-1 matrices $A = \mathbf{u}\mathbf{v}^T + \mathbf{w}\mathbf{x}^T$

$$A = \mathbf{u}\mathbf{v}^T + \mathbf{w}\mathbf{x}^T = \begin{bmatrix} | & | & \dots & | \\ v_1\mathbf{u} + x_1\mathbf{w} & v_2\mathbf{u} + x_2\mathbf{w} & \dots & v_m\mathbf{u} + x_m\mathbf{w} \\ | & | & \dots & | \end{bmatrix} = \begin{bmatrix} — & u_1\mathbf{v}^T + w_1\mathbf{x}^T & — \\ — & u_2\mathbf{v}^T + w_2\mathbf{x}^T & — \\ \vdots & \ddots & \vdots \\ — & u_n\mathbf{v}^T + w_n\mathbf{x}^T & — \end{bmatrix} = \begin{bmatrix} | & | \\ \mathbf{u} & \mathbf{w} \\ | & | \end{bmatrix} \begin{bmatrix} — & \mathbf{v}^T & — \\ — & \mathbf{x}^T & — \end{bmatrix}$$

Rank-k if it is sum of k rank-1 matrices and cannot be written as sum of $k-1$ or fewer rank-1 matrices

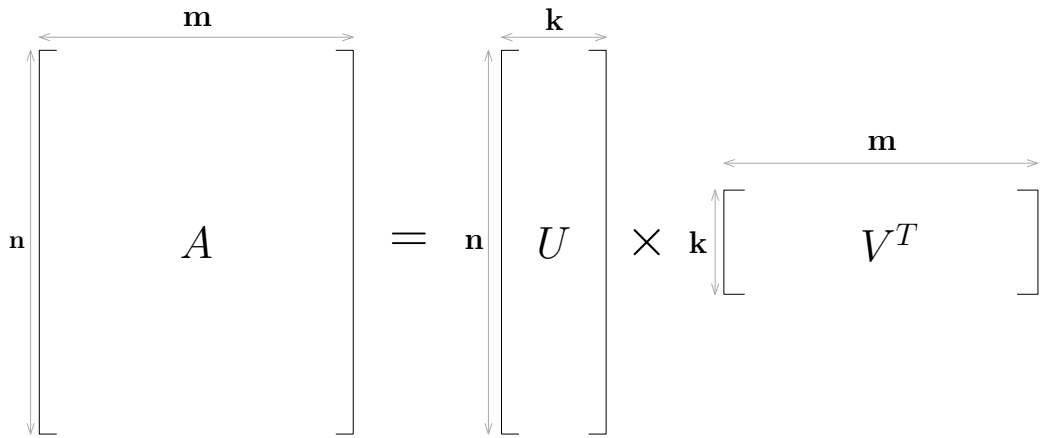
Rank Factorization of a matrix

An $n \times m$ matrix A has rank-k if A can be "factored into" the product of a

- $(n \times k)$ matrix U and ▷ tall and skinny
- $(k \times n)$ matrix V^T ▷ short and long

$$A = UV^T$$

- A cannot be factored into $n \times (k-1)$ and $(k-1) \times m$ matrices



- columns of U are the columns of the rank-1 factors $\mathbf{u}_i$'s
- rows of V^T are the rows of the rank-1 factors $\mathbf{v}_i$'s

All definitions of rank are equivalent - each implies the other

Rank Factorization of a matrix

- $n \times n$ matrix A is "full rank" if it has rank n
  - It uniquely maps $n \times 1$ vectors to $n \times 1$ vectors
  - A is a "bijection", A is invertible
- If $rank(A) < n$, then A is a singular matrix (rank deficient matrix)
  - The resulting dimensionality is $\leq n - 1$
  - Cannot get pre-images from images
  - A is not invertible
- There cannot be any inverse for a non-square matrix

Low Rank Structure in Data

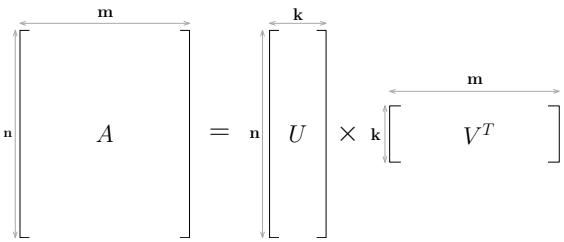
Low rank data matrix

A: a $n \times m$ data matrix ▷ rows: data points – columns: features

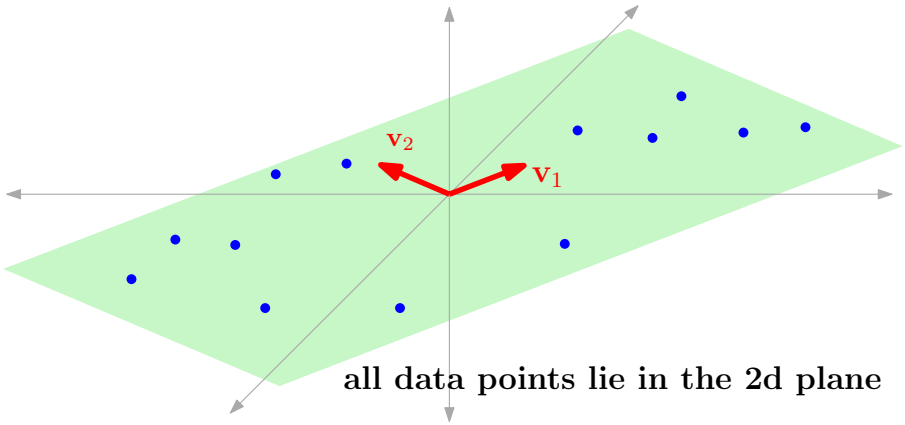
If A has rank k, then $A = UV^T$ ▷ $|U| = n \times k$ $|V| = k \times m$

Each row (data point) of A can be represented as a linear combination of $\mathbf{v}_1, \ldots, \mathbf{v}_k$

$\mathbf{a}_i = \sum_{j=1}^{k} u_{ij}\mathbf{v}_j$, u_{ij} are projection lengths of $\mathbf{a}_i$ on $\mathbf{v}_j$



Geometrically, all data lie in a k-d subspace (spanned by $\mathbf{v}_1, \ldots, \mathbf{v}_k$)



Data Compression

Space reqtt for A: $n \times m$

Store the matrix U and V

Space reqtt: $k(n + m)$

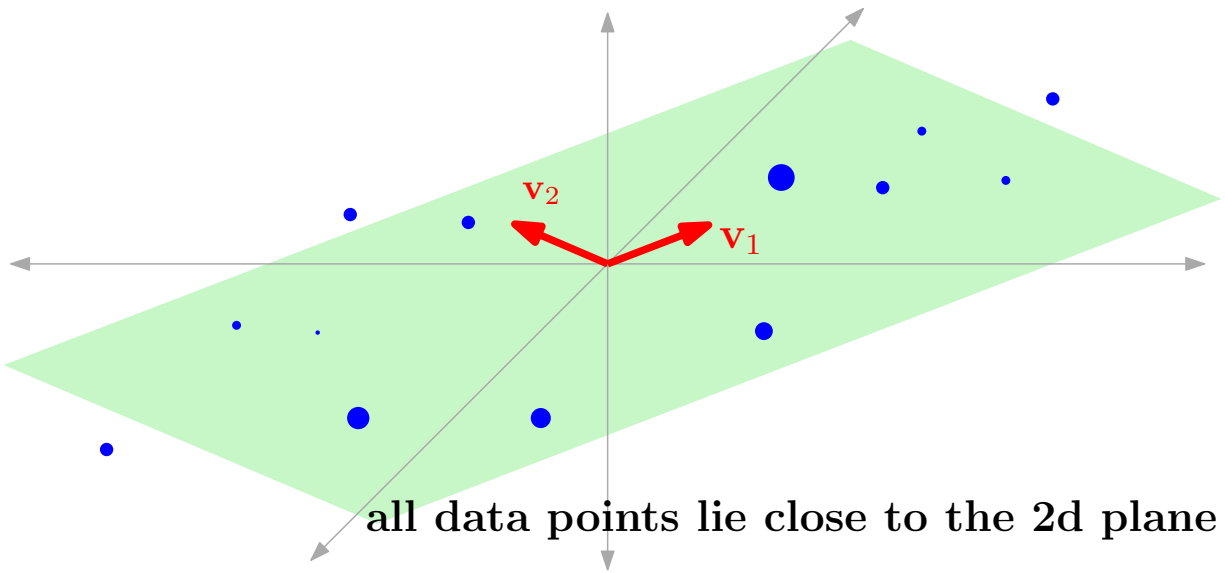
Low rank approximation

Data may not be in a k-d subspace,

but it may be lying 'close by' to a low dimensional subspace

We say data is approximately low rank

May not get $A = UV^T$ – would like to find U and V so $A \simeq UV^T$



Low rank approximation

May not get $A = UV^T$ – would like to find U and V so $A \simeq UV^T$

Need a goodness measure to assess $A \simeq UV^T$

$$\sum_{i=1}^{n} \|\mathbf{a}_i - \sum_{j=1}^{k} u_{ij}\mathbf{v}_j\|^2 =: \|A - UV^T\|_F^2$$

For a matrix M, $\|M\|_F = \sqrt{\sum_{i,j} M_{ij}^2}$ is the Frobenius norm of M

The optimization problem of finding the best low rank approximation for A

$$\underset{V \in \mathbb{R}^{k\times m},\ U \in \mathbb{R}^{n\times k}}{\arg\min} \|A - UV^T\|_F^2$$

Why expect low rank structure

Data is not necessarily described by the attributes in which it is measured

I am going to show you two examples with dependencies between columns

These examples are adapted from real-world data

Housing Data

ID	Beds	Baths	Living sq-ft	Lot sq-ft	Floors	Garage Cars	List Price	Sale Price
1	1	1	870	1100	1	0	31630	31544
2	1	1	1080	1400	1	0	35920	35916
3	2	1	1250	1500	1	0	48250	48025
4	2	1	1285	1550	1	0	48965	48738
5	2	2	1460	1800	2	1	67540	67633
6	3	2	1560	1800	1	0	68440	68763
7	3	2	1630	1900	2	1	79870	79533
8	3	2	2050	2500	2	1	88450	88054
9	3	2.5	2120	2600	2	2	102380	102576
10	4	2	2360	2800	2	1	103640	103892
11	4	2.5	2500	3000	2	1	109000	109523
12	4	2.5	2570	3100	2	1	110430	110393
13	4	3	2710	3300	3	2	125790	125945
14	5	2	2880	3400	2	2	133120	133503
15	5	2.5	2880	3400	3	2	135620	136124
16	5	2.5	3300	4000	3	2	144200	144365
17	5	3	3650	4500	3	2	153850	154444
18	5	3	3720	4600	3	3	165280	165439

Why expect low rank structure

Housing Data : Rank of this matrix is not 8 Some linear dependencies are shown (there may be others including non-linear)

$$\textbf{List-Price} = 10k\times \textbf{ bed } +5k\times\textbf{baths } +9\times \textbf{ liv-sqFT } +8\times \textbf{ Lot } +10k\times \textbf{ Cars}$$

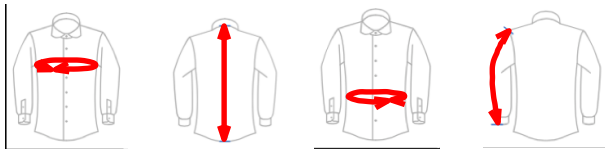
ID	Beds	Baths	Living sq-ft	Lot sq-ft	Floors	Garage Cars	List Price	Sale Price
1	1	1	870	1100	1	0	31630	31544
2	1	1	1080	1400	1	0	35920	35916
3	2	1	1250	1500	1	0	48250	48025
4	2	1	1285	1550	1	0	48965	48738
5	2	2	1460	1800	2	1	67540	67633
6	3	2	1560	1800	1	0	68440	68763
7	3	2	1630	1900	2	1	79870	79533
8	3	2	2050	2500	2	1	88450	88054
9	3	2.5	2120	2600	2	2	102380	102576
10	4	2	2360	2800	2	1	103640	103892
11	4	2.5	2500	3000	2	1	109000	109523
12	4	2.5	2570	3100	2	1	110430	110393
13	4	3	2710	3300	3	2	125790	125945
14	5	2	2880	3400	2	2	133120	133503
15	5	2.5	2880	3400	3	2	135620	136124
16	5	2.5	3300	4000	3	2	144200	144365
17	5	3	3650	4500	3	2	153850	154444
18	5	3	3720	4600	3	3	165280	165439

$$\textbf{Sale Price} = (1 \pm 0.02) \times \textbf{List Price}$$

Why expect low rank structure

Shirt Dimension Many measurements (chest and waist circumferences, sleeve and back lengths) for shirt

In market shirts are marked with collar measurement only

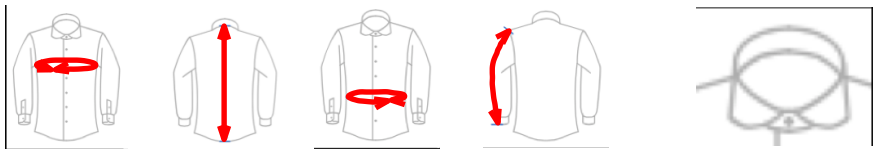


Chest	Back	Waist	Sleeve
104	81	98	67
107	81	100	67
110	82	102	67
113	82	104	67
116	83	106	68
120	83	110	68
124	84	114	68
128	84	118	68
132	85	122	68
136	85	126	68

Why expect low rank structure

Shirt Dimension The collar feature is a linear combination of other features. The data actually lies in a one dimensional space

$$\textbf{Collar} = 0.44\times \textbf{Chest} + 0.015\times \textbf{Back} - 0.2\times \textbf{Waist} + 0.153\times \textbf{Sleeve}$$



Chest	Back	Waist	Sleeve	Collar
104	81	98	67	37
107	81	100	67	38
110	82	102	67	39
113	82	104	67	40
116	83	106	68	41
120	83	110	68	42
124	84	114	68	43
128	84	118	68	44
132	85	122	68	45
136	85	126	68	46

Singular Value Decomposition

Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) is a fundamental matrix factorization technique used in recommendation systems

- SVD decomposes the user-item matrix into three smaller matrices that capture latent relationships between users and items
- These latent factors represent hidden preferences or features that are not directly observable
- By approximating the original matrix with lower-dimensional matrices, SVD can make accurate predictions for missing values

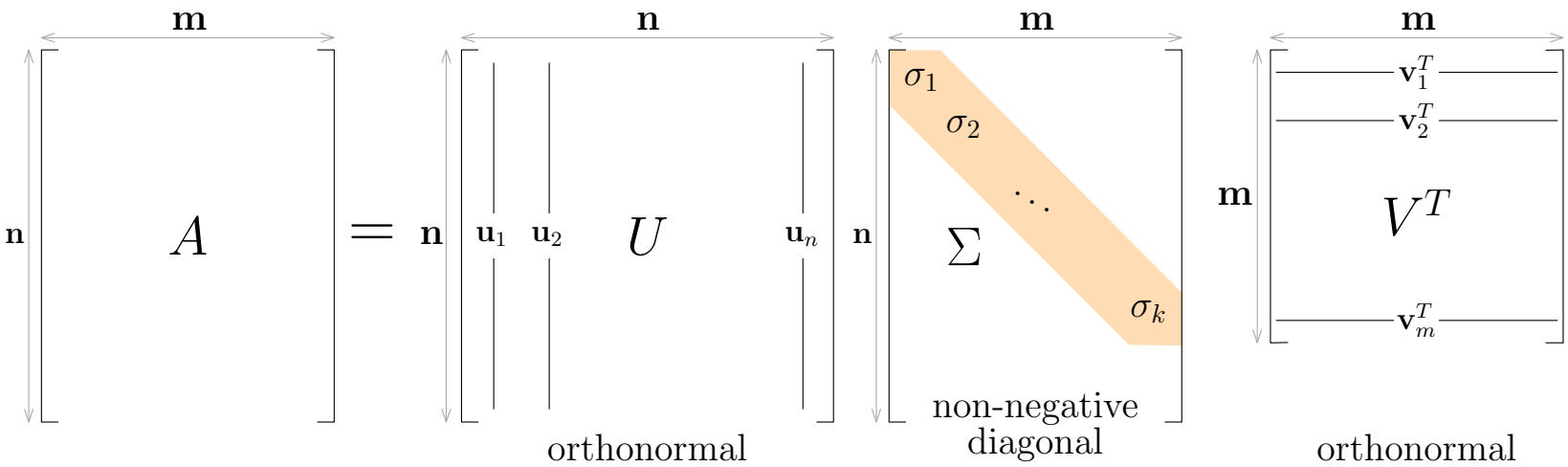
Singular Value Decomposition

Theorem

Any $n \times m$ matrix can be written as a product of three matrices

$$A = U \Sigma V^T$$

- *U is a $n \times n$ orthogonal matrix (columns are orthonormal)*
- *V is a $m \times m$ orthogonal matrix*
- *Σ is a $n \times m$ diagonal matrix, with non-negative entries and entries at the main diagonal are sorted from highest value to lowest*



Singular Value Decomposition

$$\text{SVD:} \qquad A = U\Sigma V^T$$

- A is the original user-item matrix
- U is orthogonal – its columns are called left singular vectors
  - ▷ Columns of U are user latent factors
- V is orthogonal – its columns are called right singular vectors
  - ▷ Columns of V^T are item latent factors
- Diagonal entries of Σ are called singular values
  - ▷ singular values represent the strength of each latent factor

SVD reveals latent factors corresponding to hidden relationships between users and items

SVD projects both rows and columns of A into this shared latent feature space, allowing missing values prediction based on proximity of rows and columns in this space

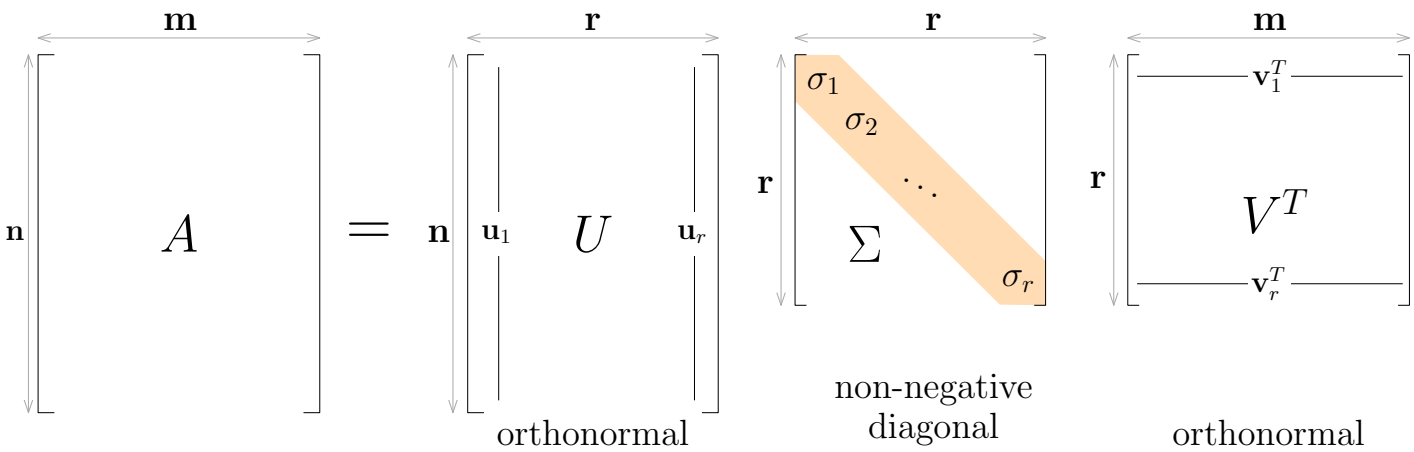
Compact Singular Value Decomposition

Keeping only the top r singular values and their corresponding vectors, we approximate A using a low-rank matrix

Theorem (Compact SVD)

Any $n \times m$ matrix with rank $r \leq \min\{m, n\}$ can be written as a product of three matrices, $A = U\Sigma V^T$

- *U is a $n \times r$ orthogonal matrix (columns are orthonormal)*
- *V is a $r \times r$ orthogonal matrix*
- *Σ is a $r \times m$ diagonal matrix, with non-negative entries and entries at the main diagonal are sorted from highest value to lowest*

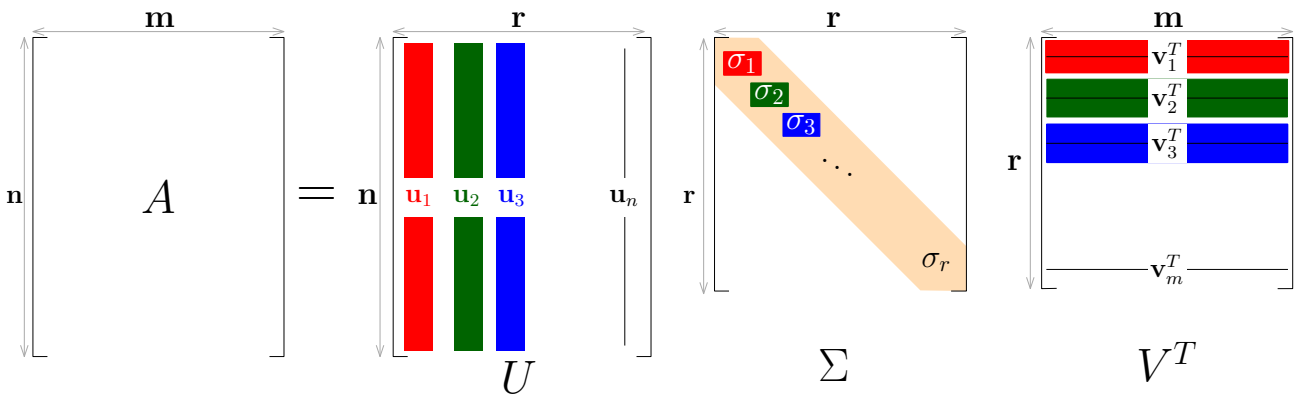


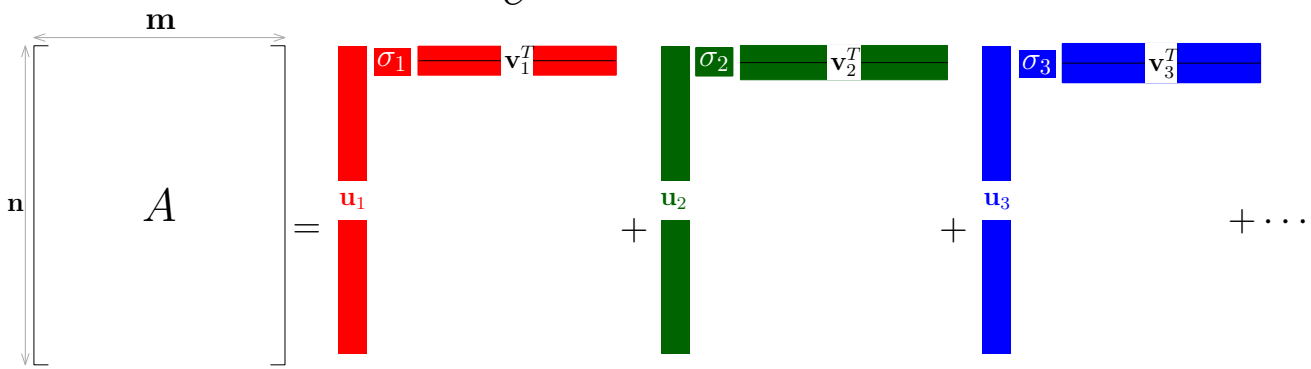
Spectral decomposition of a matrix

From SVD of $A \in \mathbb{R}^{n \times m}$ we can get spectral decomposition of A

i.e. Express A as linear combination of r rank-1 matrices (outer products of singular vectors) – coefficients are the corresponding singular values

$$A = U\Sigma V^T \quad \Leftrightarrow \quad A = \sum_{\ell=1}^{\min\{m,n\}} \sigma_\ell \, \mathbf{u}_\ell \circ \mathbf{v}_\ell^T$$



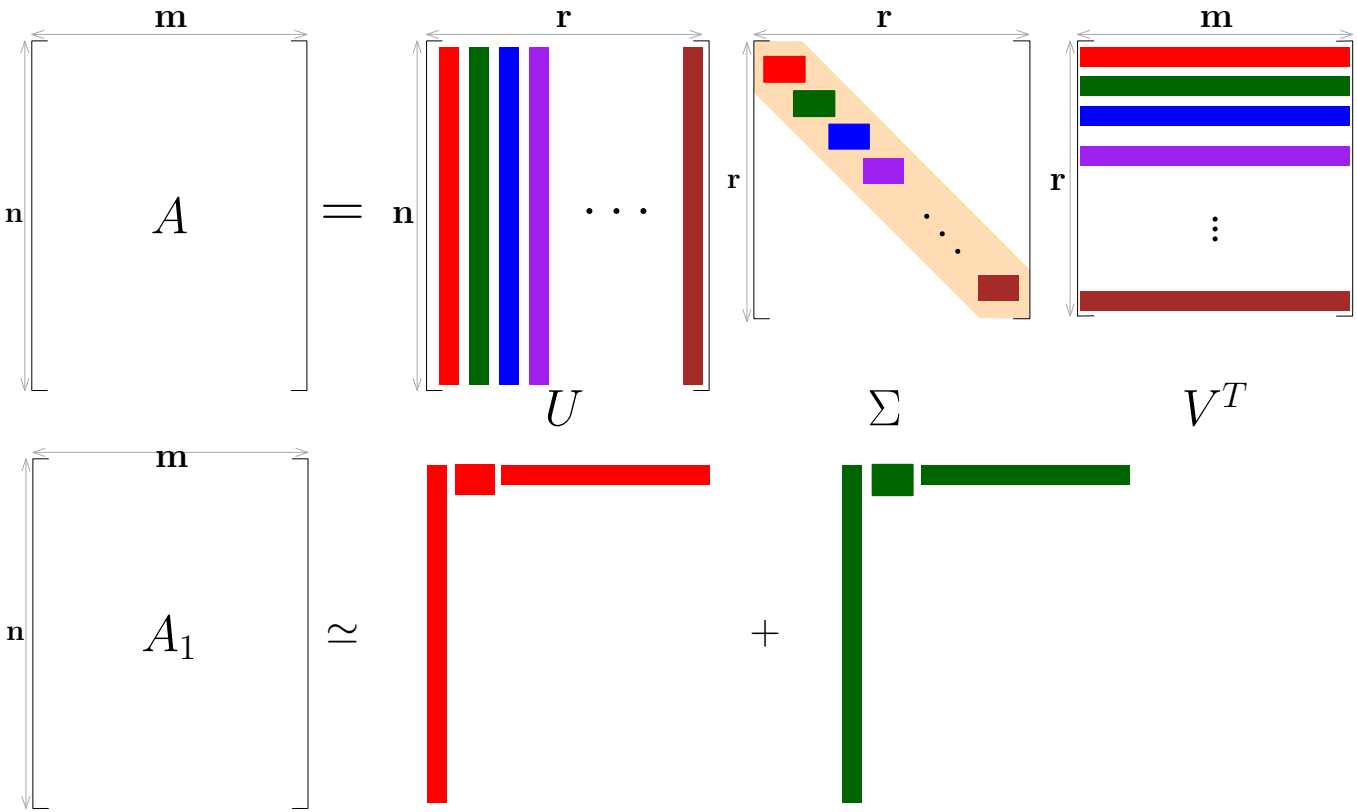


Truncated SVD

$$A = U\Sigma V^T$$

$$A_k = \sum_{\ell=1}^{k} \sigma_\ell \mathbf{u}_\ell \circ \mathbf{v}_\ell^T + \sum_{\ell=k+1}^{r} \sigma_\ell \mathbf{u}_\ell \circ \mathbf{v}_\ell^T$$

$$A = \sum_{\ell=1}^{r} \sigma_\ell \, \mathbf{u}_\ell \circ \mathbf{v}_\ell^T$$

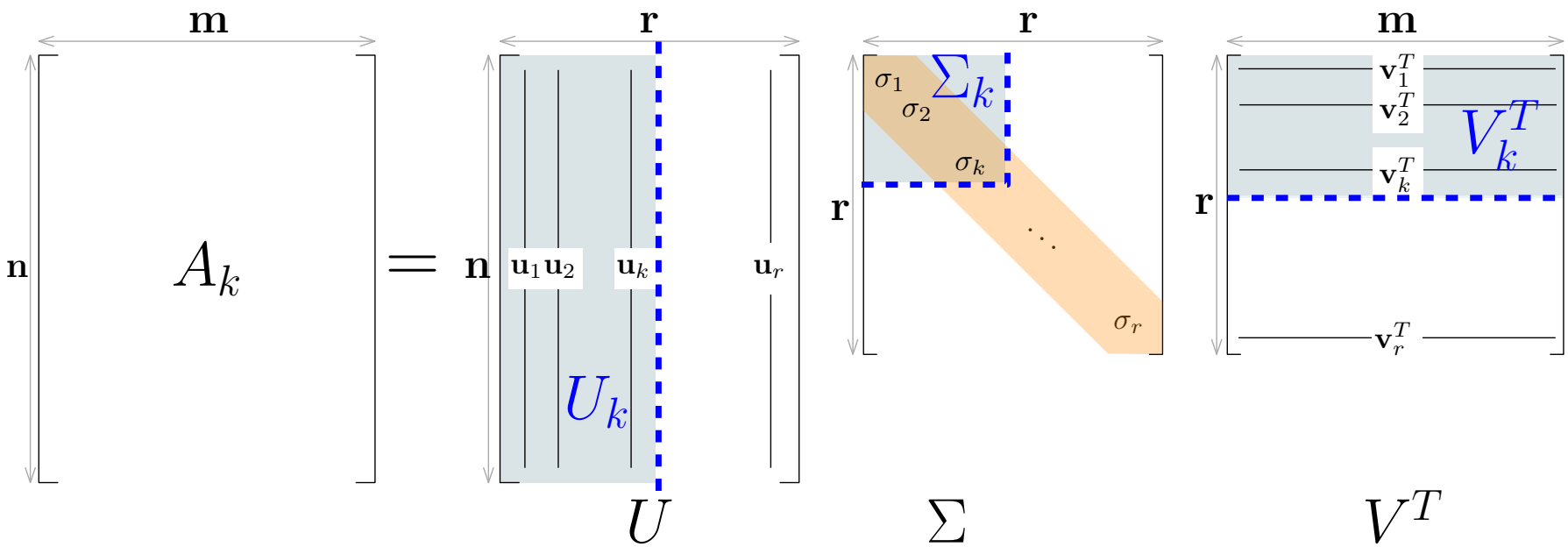


Set to 0 (truncate) the last $r - k$ singular values (σ_{k+1} to σ_r)

Truncated SVD

- $U_k \in \mathbb{R}^{n \times k}$: the first k left singular vectors (the first k columns of U)
- $\Sigma_k \in \mathbb{R}^{k \times k}$: the first k singular values
- V_k^T be the first k right singular vectors

$$A_k = \sum_{\ell=1}^{k} \sigma_\ell \mathbf{u}_\ell \circ \mathbf{v}_\ell^T + \cancel{\sum_{\ell=k+1}^{r} \sigma_\ell \mathbf{u}_\ell \circ \mathbf{v}_\ell^T} = U_k \Sigma_k V_k^T$$



Truncated SVD: Low-Rank Approximation of a Matrix

Truncated SVD: $A \approx U_k \Sigma_k V_k^T$

- A is the originalmatrix
- U_k contains the top k columns of U vectors
  - ▷ Columns of U are user latent factors
- V_k contains the top k rows of V^T
  - ▷ Columns of V^T are item latent factors
- Σ_k is the diagonal matrix with the top k singular values
  - ▷ singular values represent the strength of each latent factor

This low-rank approximation captures the most important latent factors while ignoring less significant information

Truncated SVD based Factorization of user-item Matrix

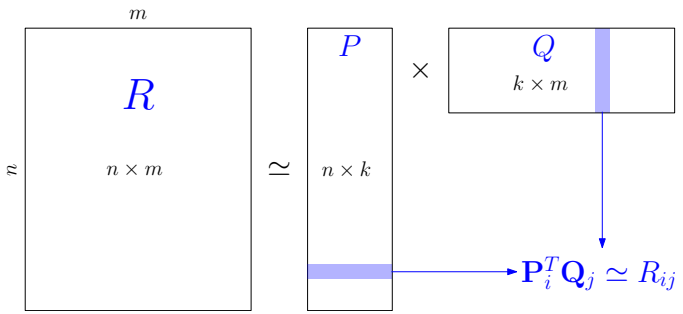
Finding the best factorization of the rating matrix, i.e., $R \simeq PQ$

$$\min_{\substack{P \in \mathbb{R}^{n \times k} \\ Q \in \mathbb{R}^{m \times k}}} \sum_{(i,j)} \left(R_{ij} - P_i Q_j^T \right)^2$$

Truncated SVD: $R_k \approx U_k \Sigma_k V_k^T$

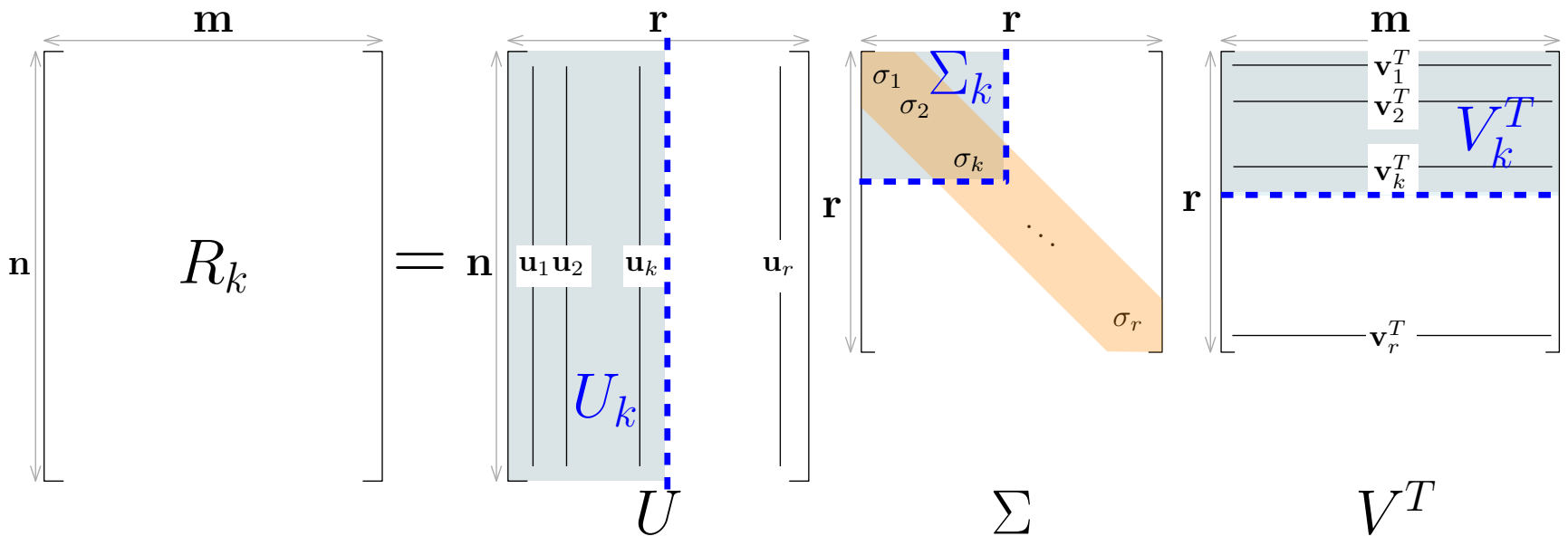
Theorem: R_k is the best rank-k approximation to R

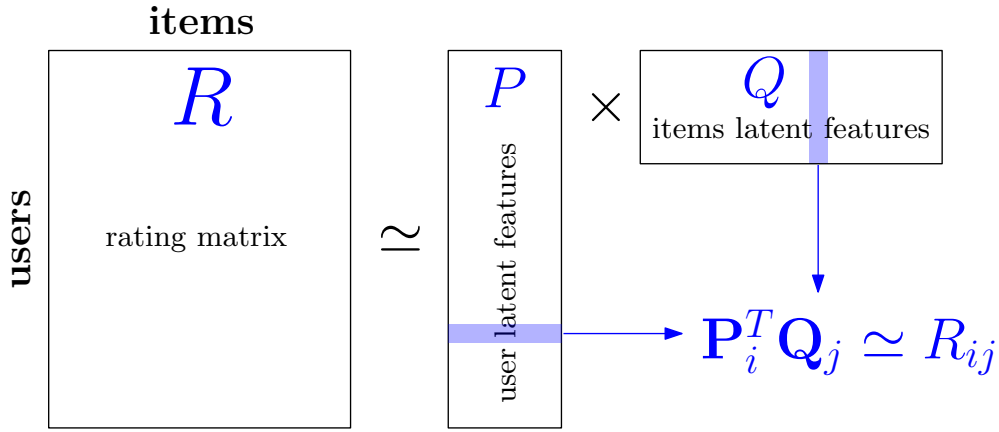
- the P in the above problem would be $U_k \sqrt{\Sigma_k}$
- and Q would be $\sqrt{\Sigma_k} V_k^T$



SVD Application: Recommenders

Using SVD to get $R = PQ$





$$\mathbf{P} = U_k \sqrt{\Sigma_k}$$

$$\mathbf{Q} = \sqrt{\Sigma_k}\, V_k^T$$

SVD Application: Recommenders

- SVD is not the best approach to factorize rating matrix
- Typically R will have many values missing
- SVD will adjust U, Σ and V to the 0's or any default values
- One can try other default values
  - matrix average, row averages, column averages, ANOVA
- SVD performs good if R is close to rank-k and has few missing values

Alternating Least Squares (ALS)

Alternating Least Squares (ALS) is an optimization algorithm for matrix factorization

ALS decomposes the user-item interaction matrix into two low-rank matrices; one for users and one for items

ALS alternates between fixing one set of variables (either user or item latent factors) and solving for the other

- Scalability and Parallelizability: ALS can handle large datasets and is often parallelized to improve performance.
- Iterative Optimization: ALS alternates between solving for user and item matrices in a step-by-step manner.
- Convergence: ALS guarantees convergence to a local minimum in a fixed number of iterations.

Alternating Least Squares (ALS)

ALS aims to minimize the following objective function:

$$\arg\min_{P,Q} \underbrace{\sum_{(i,j)\in\mathcal{D}} (r_{ij} - P_i^T Q_j)^2}_{\text{error term}} + \underbrace{\lambda \left(\|U\|^2 + \|V\|^2 \right)}_{\substack{\text{regularization term} \\ \text{avoids overfitting}}}$$

- r_{ij} is the observed rating for user i and item j
- U_i and V_j are the user and item latent factors, respectively
- λ is the regularization parameter to prevent overfitting
- $\mathcal{D}$ is the set of observed user-item interactions

ALS alternates between user and item matrices optimizing prediction error

1. Fixing Item Matrix: ALS first fixes the item latent factors and solves for the user latent factors by minimizing the least squares objective
2. Fixing User Matrix: Next, it fixes the user latent factors and solves for the item latent factors
3. Repeat until convergence, minimizing the prediction error at each step

Hybrid Methods for Recommendation Systems

Hybrid recommenders combine multiple models to leverage the strengths of different approaches while mitigating their shortcomings

- Content-Based Filtering: Recommends based on item features, but lacks diversity and generalization
- Collaborative Filtering: Leverages collective user preferences but struggles with the cold start problem
- Matrix Factorization: Captures latent factors, but can overfit without sufficient regularization

- Improving Accuracy: overcome the limitations of individual models, (e.g., cold start, scalability)
- Handling Diverse Data: Different types of data (e.g., explicit ratings, implicit feedback, item features) can be effectively utilized by different models within a hybrid system
- Increasing Coverage: Hybrid systems can recommend items to more users by addressing the shortcomings of models in isolation

Weighted Hybrid

In a weighted hybrid system, the predictions from different models are combined using a predefined or learned weighting scheme:

$$\hat{r}_{ui} = \alpha_1 \hat{r}_{ui}^{(model_1)} + \alpha_2 \hat{r}_{ui}^{(model_2)} + \cdots + \alpha_n \hat{r}_{ui}^{(model_n)}$$

Where α_i are the weights assigned to each model, representing their contribution to the final prediction. This allows the system to balance the influence of each model based on its effectiveness

- Weights can be fixed or learned through techniques like cross-validation
- CF might be assigned a higher weight for users with sufficient history, while content-based filtering might be emphasized for new users.

Switching Hybrid

In a switching hybrid system, the algorithm dynamically switches between different recommendation models based on the context:

- For example, for users with rich interaction histories, the system might use collaborative filtering, but for new users, content-based filtering is used.
- The decision to switch models can be based on specific conditions like the amount of user data available or the type of item being recommended.

Switching hybrids allow for flexible model selection, adapting to the needs of the user or dataset.

Mixed Hybrid

A mixed hybrid system integrates features from multiple models and processes them simultaneously:

- The recommendations from collaborative filtering, content-based filtering, and other methods are combined and presented as a unified recommendation.
- This allows for more diverse recommendations, drawing on the strengths of multiple models in parallel.

For example, in a music recommendation system, a mixed hybrid might recommend songs based on both user listening history (collaborative filtering) and song characteristics (content-based filtering).

Feature Combination

Feature combination merges the feature spaces of different models, such as content-based and collaborative filtering, to build a more comprehensive recommendation model:

- The feature spaces from multiple models are combined to create richer user and item profiles.
- This method allows for more accurate predictions by considering both user-item interactions and item content features.

For example, merging user behavior data with item metadata can improve the accuracy of movie recommendations by combining user preferences with genre and director information.

Cascade Hybrid

In a cascade hybrid system, algorithms are layered sequentially, with each model refining the recommendations of the previous model:

- A first model generates an initial list of recommendations, which a second model then refines by applying additional filtering or ranking criteria.
- For example, collaborative filtering might generate an initial recommendation list, which is then refined using content-based filtering to prioritize items that align with user preferences.

This method allows each model to specialize in different aspects of the recommendation process.

Feature Augmentation

Feature augmentation enhances the user or item profiles by incorporating external features:

- External data sources, such as user demographics or social media activity, are added to the feature set.
- These additional features help improve the accuracy of recommendations by providing more context about user preferences or item characteristics.

For example, in an e-commerce recommendation system, user demographics or browsing history can be used to refine product recommendations beyond basic interaction data.

Meta-Level Hybrid

In a meta-level hybrid system, the output of one model is used as the input to another model:

- For instance, the latent factors from a matrix factorization model might be used as features in a content-based filtering model.
- This allows one model to leverage the learned structure or patterns of another model, leading to more accurate predictions.

For example, in a movie recommender, a matrix factorization model might first generate latent factors that are then used to enhance the predictions of a content-based model.

The Netflix Challenge

The Netflix Challenge (aka the Netflix Prize) was a multi-year competition with an ultimate prize of $1,000,000 for achieving a 10% improvement in the accuracy of Netflix's recommendation algorithm

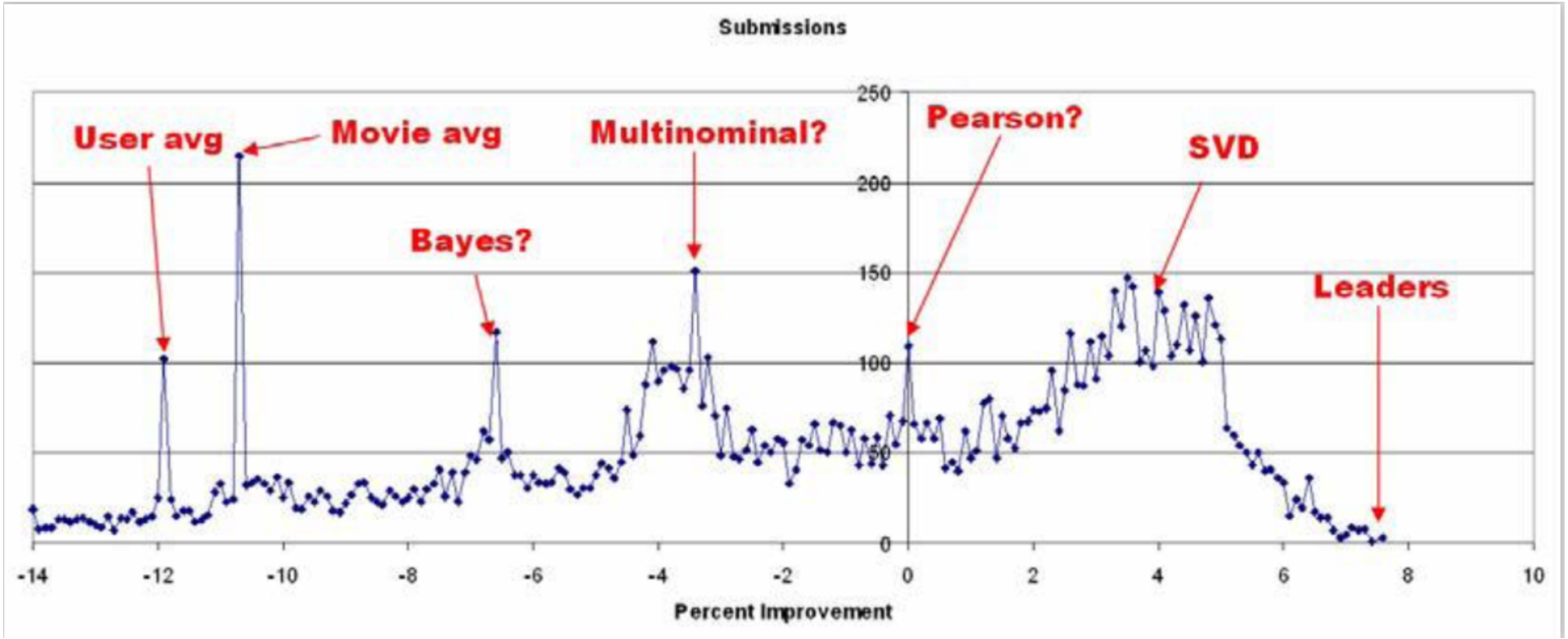
- Launch (2006): Public release of data and competition guidelines.
- Leaderboards: Teams could submit predictions and see their rank
- Final Prize (2009): BellKor's Pragmatic Chaos team submits the winning solution with a 10.06% improvement
- The competition attracted over 50,000 participants across more than 40,000 teams

Data Sets Used in the Netflix Challenge

For the competition, Netflix provided multiple datasets derived from actual user interaction data on the platform:

- Public Training Set: User ratings for movies
  - ▷ $\sim 1M$ ratings (480,189 users, 17,770 movies) of the form $\langle$user, movie, date of grade, grade $\in \{1, 2, 3, 4, 5\}$ $\rangle$,
- Public Probe Set For teams to validate their models on known data
- Qualifying Data Set: 2,817,131 triplets, $\langle$user, movie, date$\rangle$
  - ▷ grade was known to jury only
- Hidden Quiz Set: Used to assess model performance on unseen data and helped rank teams on the public leaderboard
  - ▷ $1,408,342$ ratings, teams were told accuracy of submitted models
- Hidden Test Set: Used to rank submissions and determine winner
  - ▷ $1,408,789$ ratings

Netflix Challenge Methods



Temporal Dynamics in Recommendation Systems

User preferences are not static—they evolve over time.

Recommenders account for temporal dynamics and changes users' tastes to ensure that recommendations remain relevant

- Time Decay Models: More recent interactions are given greater weight, as they are more reflective of current user preferences.
- Seasonality: Certain preferences may change cyclically (e.g., preferences for holiday-themed movies during December).
- Trends: Users may follow trends that cause their preferences to shift over time (e.g., following a new fashion or music trend).
- Time-Based Matrix Factorization: Incorporates time as a factor in models, allowing for dynamic shifts in user preferences

In a streaming service, the system might prioritize movies a user recently watched over those watched long ago when making recommendations

Modeling Temporal Effects

Temporal Effects: Users' preferences change over time, and items may gain or lose popularity

Temporal models incorporating temporal effects into recommenders to capture time-based trends

- Dynamic Preferences: Temporal models capture shifts in user preferences, reducing prediction error for time-sensitive items
- Seasonality: Certain preferences may change cyclically, such as increased demand for specific products during holidays

  ▷ e.g., a user who purchased winter clothing last year may be recommended similar items as winter approaches again

- Time Decay Models: More recent user interactions are given greater weight, reflecting current preferences
- Regularization: Time-based regularization can prevent overfitting to older interactions

Context-Aware Recommendation Systems

Context-aware recommenders aim to improve the relevance of recommendations by considering the contextual information that affects a user's preferences

- Time: User preferences may vary throughout the day or week
- Location: The physical location of a user can influence their preferences (e.g., recommending local restaurants)
- Device: The device used (e.g., mobile, desktop) can influence the type of recommendation
- Social Context: Recommendations may vary based on whether the user is alone or with friends

Incorporating Contextual Information in Recommendations

Contextual information can be integrated into recommendation systems in various ways:

- Explicit Context: Directly collecting context data from the user, such as asking for location or activity
- Implicit Context: Inferring context from user behavior (e.g., browsing patterns, timestamps, GPS data)
- Hybrid Approaches: Combining explicit and implicit context to enhance recommendation accuracy

Example: A music streaming app may recommend different songs based on whether the user is at the gym (energetic music) or relaxing at home (calmer music)

Context-Aware Collaborative Filtering

Context-aware collaborative filtering integrates contextual information into the user-item interaction matrix through various approaches

- Multidimensional Matrix: Instead of a two-dimensional matrix (user-item), context-aware systems use a multidimensional matrix (user-item-context), adding another axis for contextual variables
- Factorization Methods: Matrix factorization techniques like SVD are extended to include context, decomposing the user-item-context matrix into lower-dimensional latent factors
- Tensor Factorization: Treating the user-item-context interactions as a tensor and applying tensor factorization techniques for recommendation

Deep Learning in Recommenders: Neural Collaborative Filtering

Neural Collaborative Filtering (NCF) applies deep neural networks to model the interactions between users and items.

Traditional collaborative filtering assumes linear interactions, NCF can captures complex and nonlinear relationships

- Model Structure: The model consists of embedding layers for users and items, followed by multiple fully connected layers that learn user-item interactions.
- Loss Function: Commonly uses binary cross-entropy or mean squared error (MSE) to optimize the recommendation.

Deep Learning in Recommenders: Autoencoders

Autoencoders are a class of neural networks used for dimensionality reduction and feature learning in recommendation systems

Autoencoders can capture user preferences and item characteristics, making them effective in collaborative filtering tasks

- Encoder: Compresses the input data (e.g., user interactions) into a lower-dimensional latent space
- Decoder: Reconstructs the input data from the compressed representation, learning useful latent features

Deep Learning in Recommenders: Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are used in recommendation systems to model sequential data, such as user interaction histories.

RNNs can capture patterns over time, making them ideal for personalized recommendations that evolve based on user behavior

- Sequential Data: RNNs can handle time-dependent data, such as the order in which items were consumed (e.g., movies, songs).
- Hidden States: RNNs maintain a hidden state that captures the sequence history, allowing the network to make predictions based on past interactions.

Privacy and Fairness in Recommendation Systems

Protecting user privacy and ensuring fairness are increasingly critical concerns in modern recommendation systems:

- Privacy:
  - Data Sensitivity: User interactions, preferences, and personal information are often collected and stored, making them vulnerable to misuse or breaches
  - Data Sharing: Recommenders sometimes require sharing data across platforms or with third parties, increasing the risk of exposure
  - User Anonymity: Preserving user anonymity while providing personalized recommendations can be difficult
- Fairness:
  - Bias Amplification: Recommendations can reinforce existing biases if data is biased, leading to unfair treatment of certain users or items
  - Underrepresentation: Minority groups or niche items may receive fewer recommendations, leading to reduced visibility and engagement
  - Equitable Treatment: Fairness ensures that all user groups receive balanced and relevant recommendations without systemic bias

Techniques for Ensuring Privacy

Techniques to protect user privacy in recommendation systems:

- Differential Privacy: Ensures that the inclusion or exclusion of a single user's data does not significantly affect the outcome of the recommendation system
  - Noise Addition: Random noise is added to the user data or query results, masking individual contributions
  - Privacy Budget: Defines the amount of information that can be learned about any individual, limiting data exposure over multiple queries
- Federated Learning: Allows models to be trained across multiple devices without sharing user data, keeping the data localized while aggregating model updates
- Data Encryption: Encrypting user interactions and data to ensure that sensitive information remains protected during processing and storage

Ensuring Fairness and Avoiding Biases

To ensure fairness in recommendation systems, several strategies can be employed:

- Fairness-Aware Algorithms: Incorporate fairness constraints directly into the recommendation algorithm, ensuring balanced treatment across different user groups
- Bias Detection: Regularly monitor for biases in the data or the recommendations and correct for underrepresented groups or items
- Diversity and Equity: Implementing diversity-promoting mechanisms that ensure a wide range of items and user groups are recommended fairly