Recommendation Systems

Predictive Analytics

Simon Business School

Mostly based on: (other sources are cited in the last slide)

- Chapter 13 in “The analytics edge” by Bertsimas, O`Hair, and Pulleyblank
- Collaborative Filtering Recommender Systems By Michael D. Ekstrand, John T. Riedl and Joseph A. Konstan

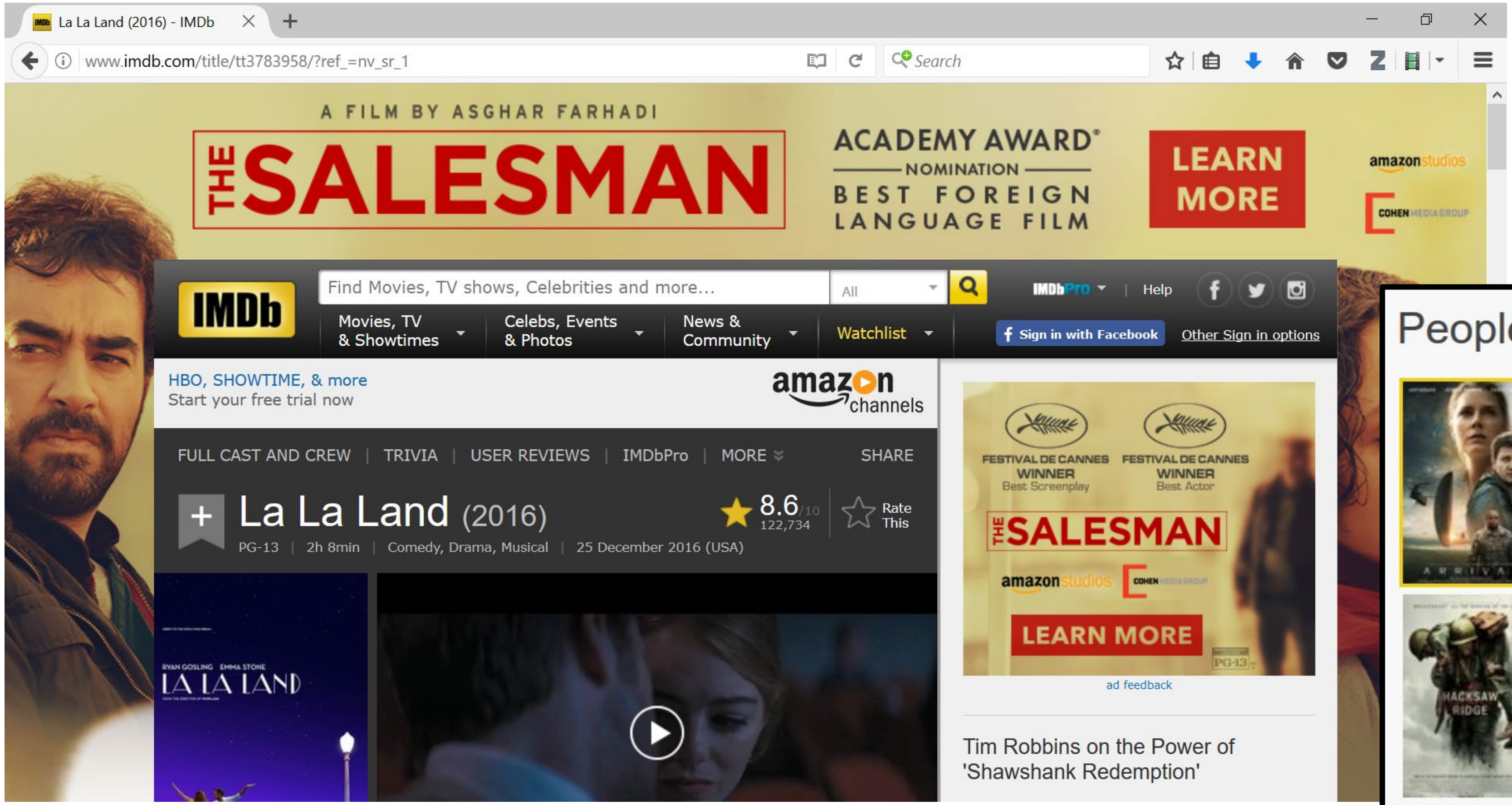
Agenda

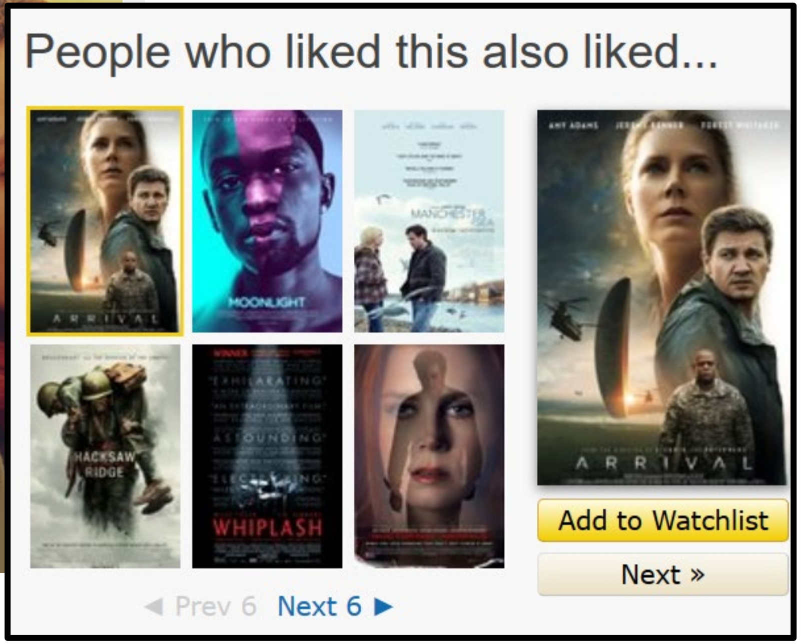
- What are recommendation (recommender) systems?
- What are their applications?
- Why are they interesting and important?
- How do they work?
- How to construct recommendation systems using Python?

What are recommendation systems?

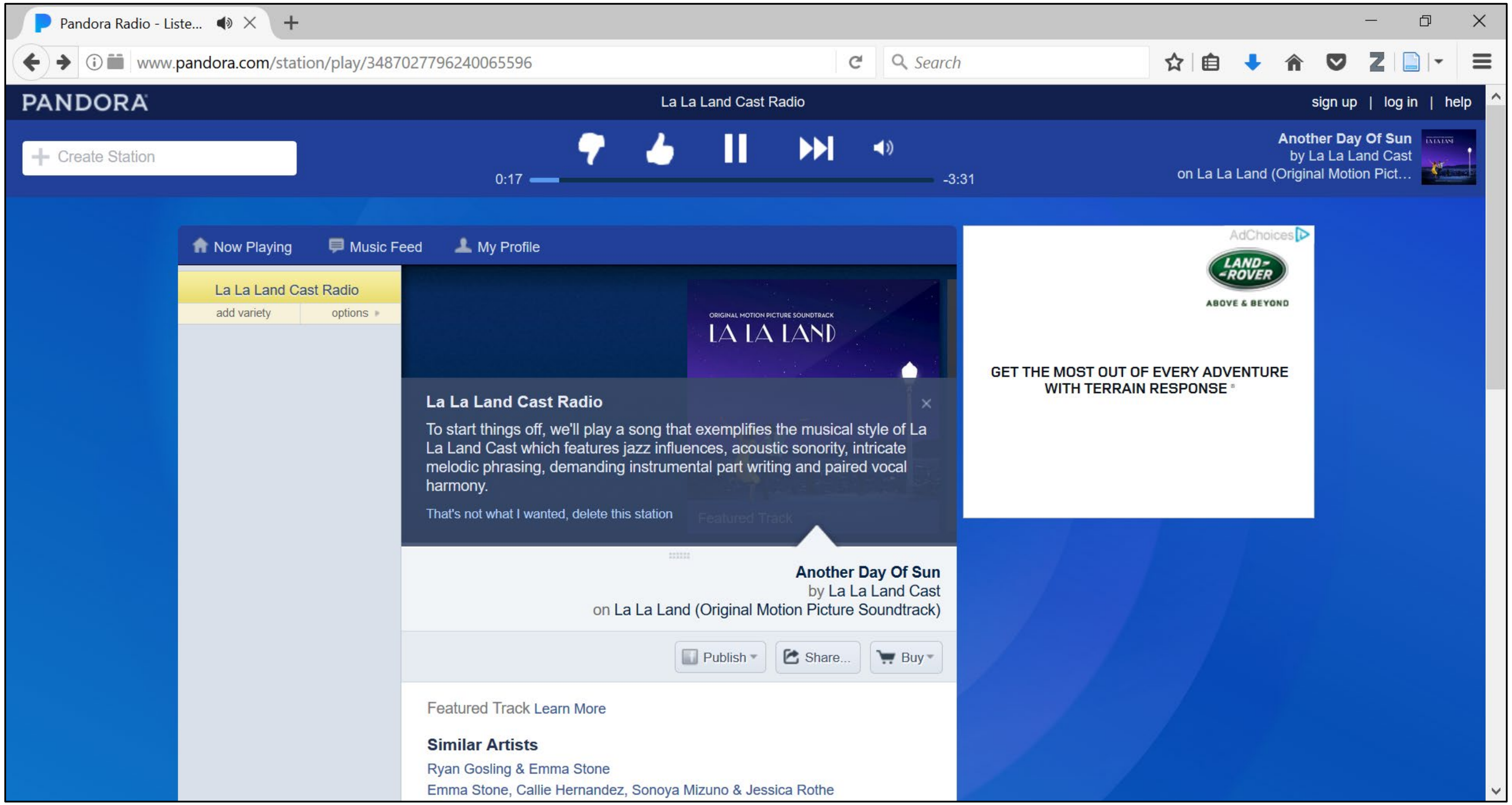
- “Systems that estimate users’ preference on **items** and proactively recommend items that **users** might like” (Zhang et al. 2019)

Example: Movie recommendation

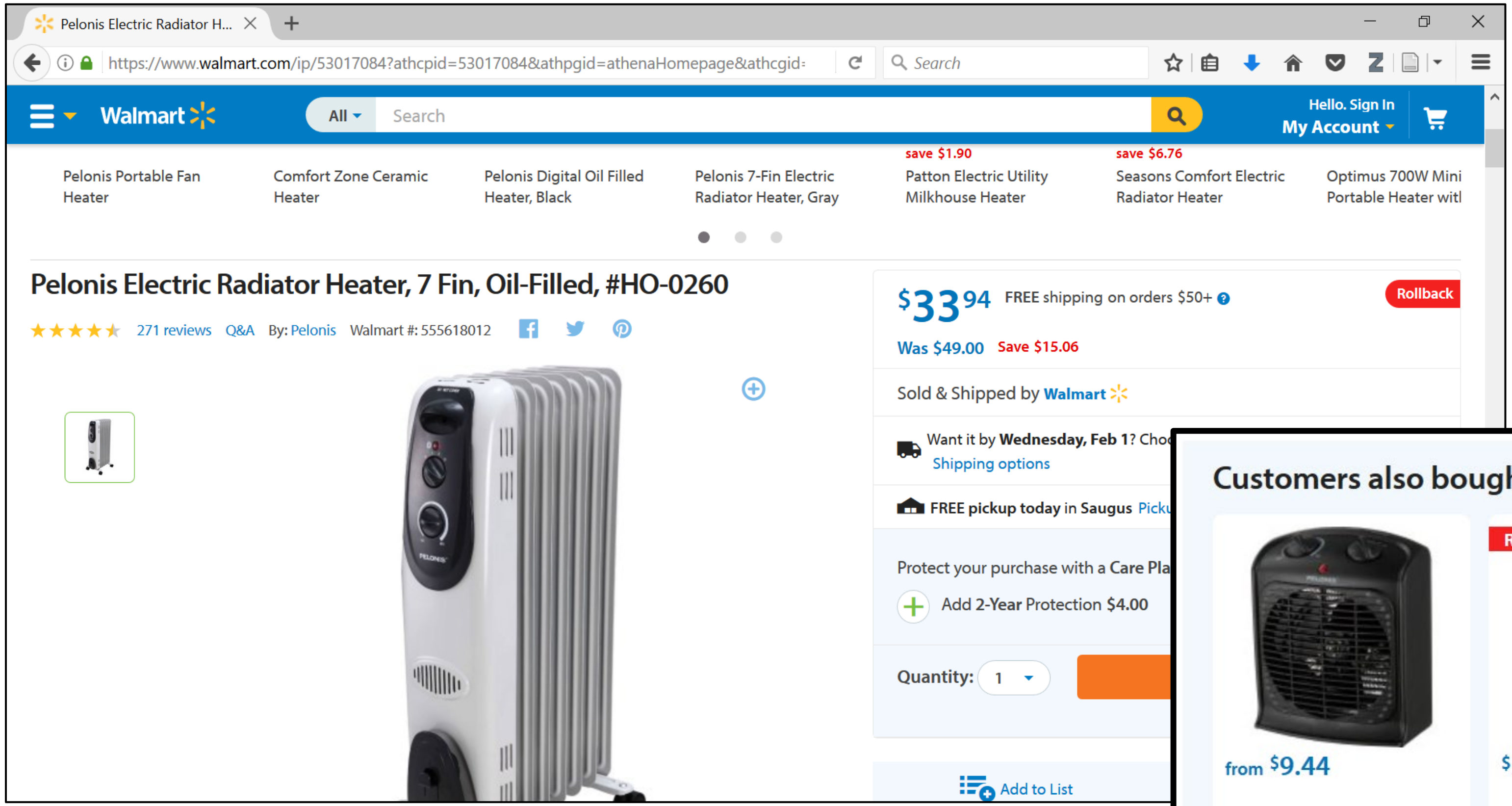


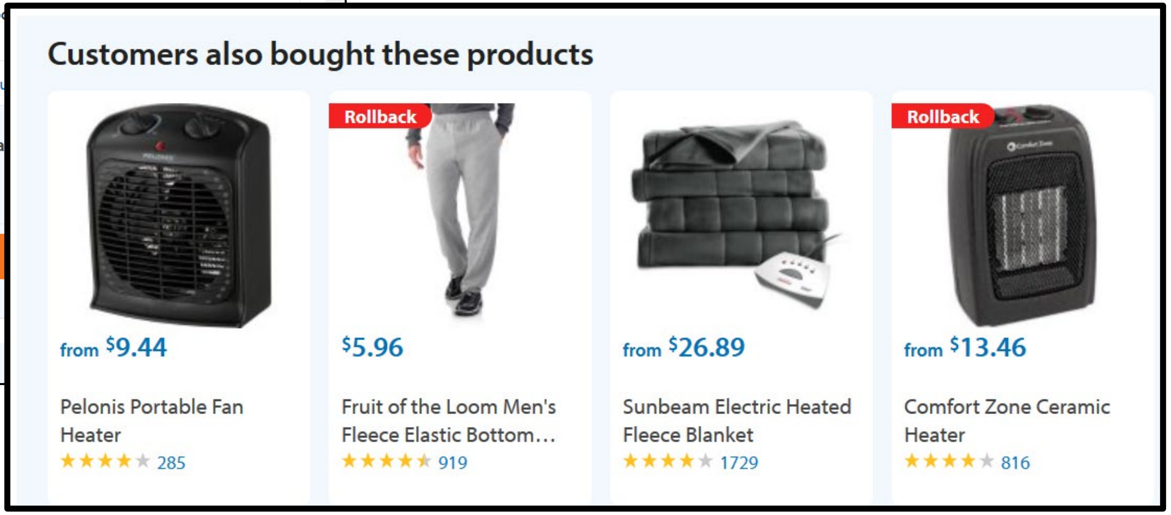


Example: Music recommendation

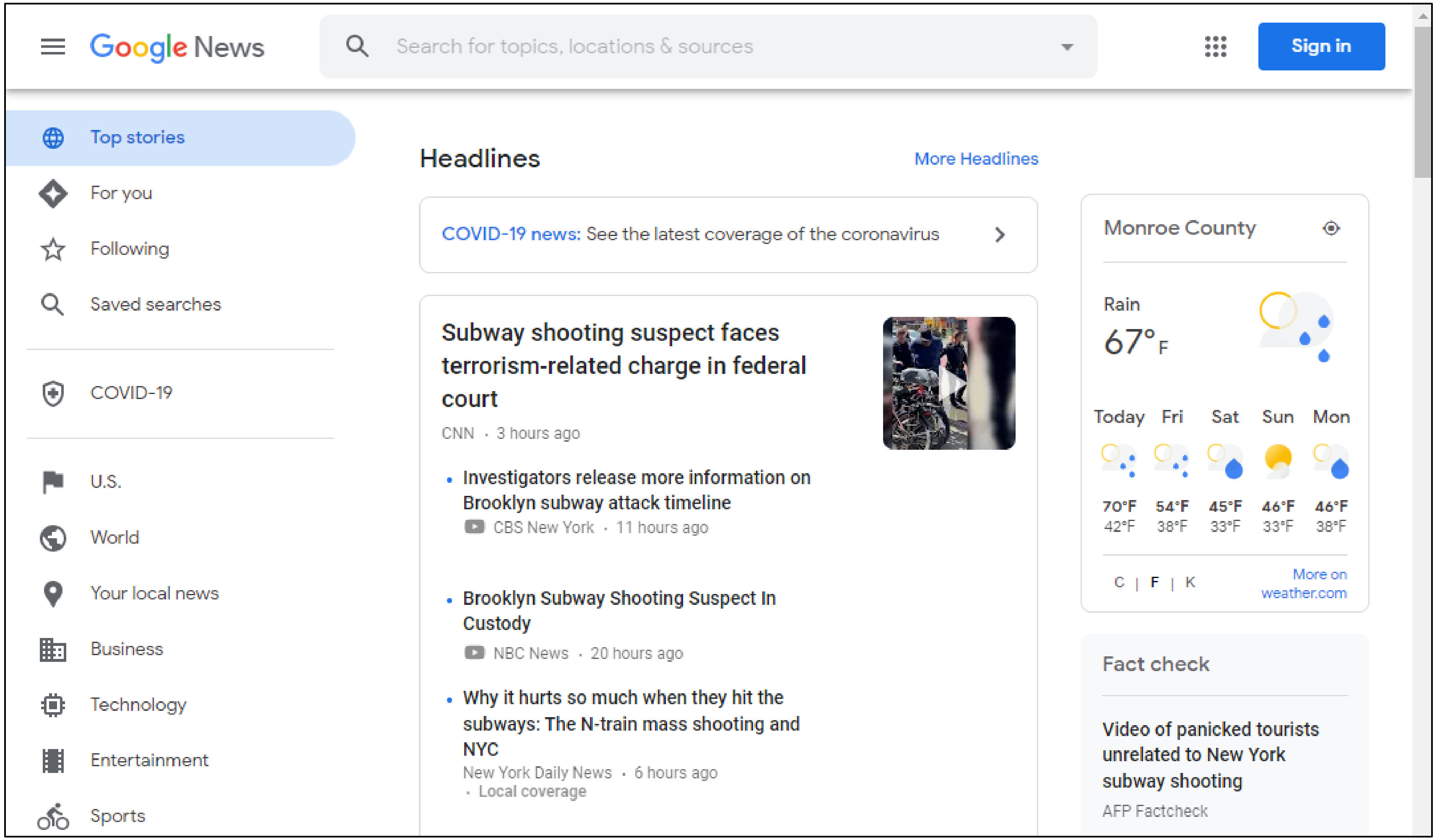


Example: E-commerce

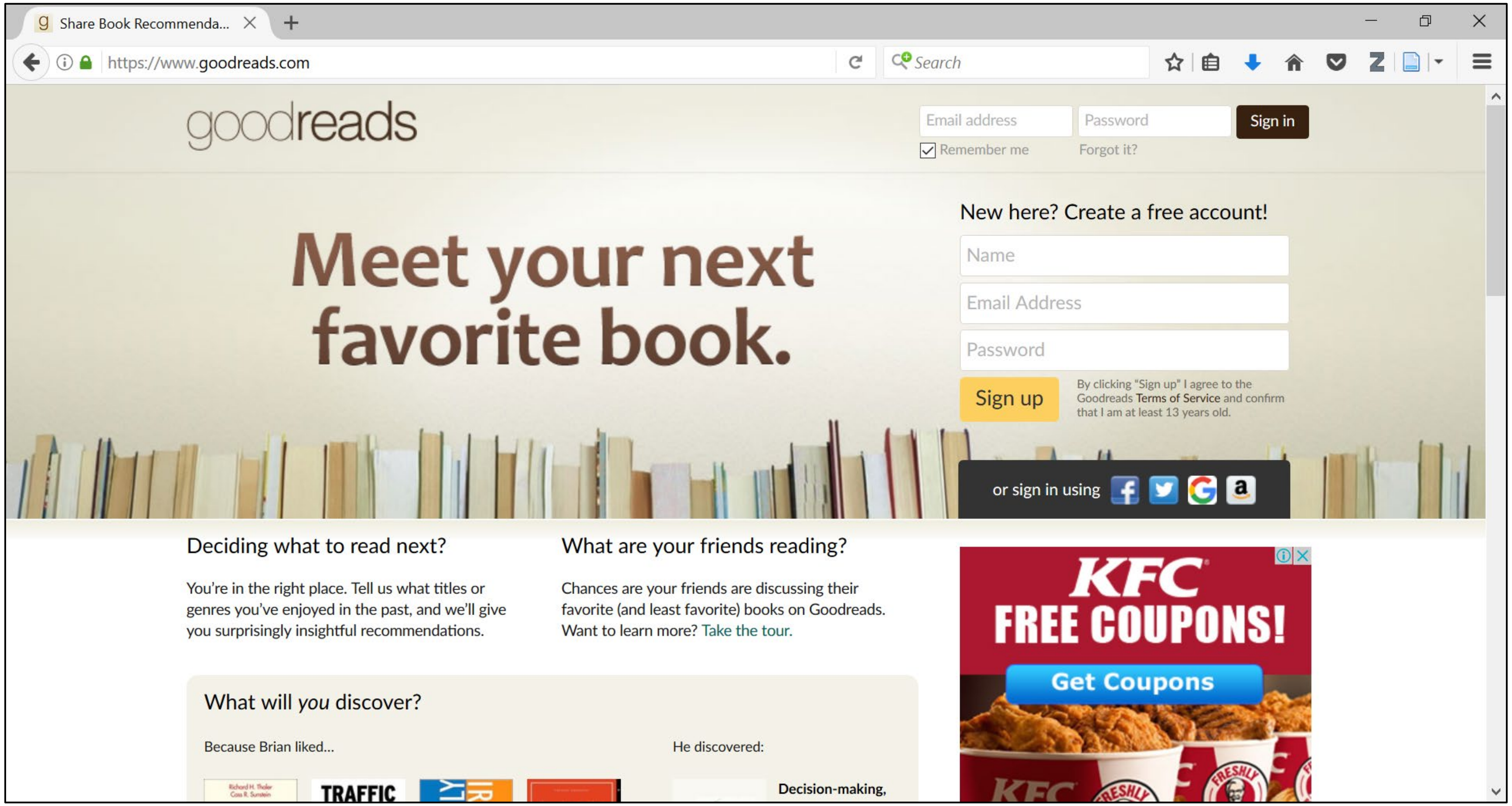




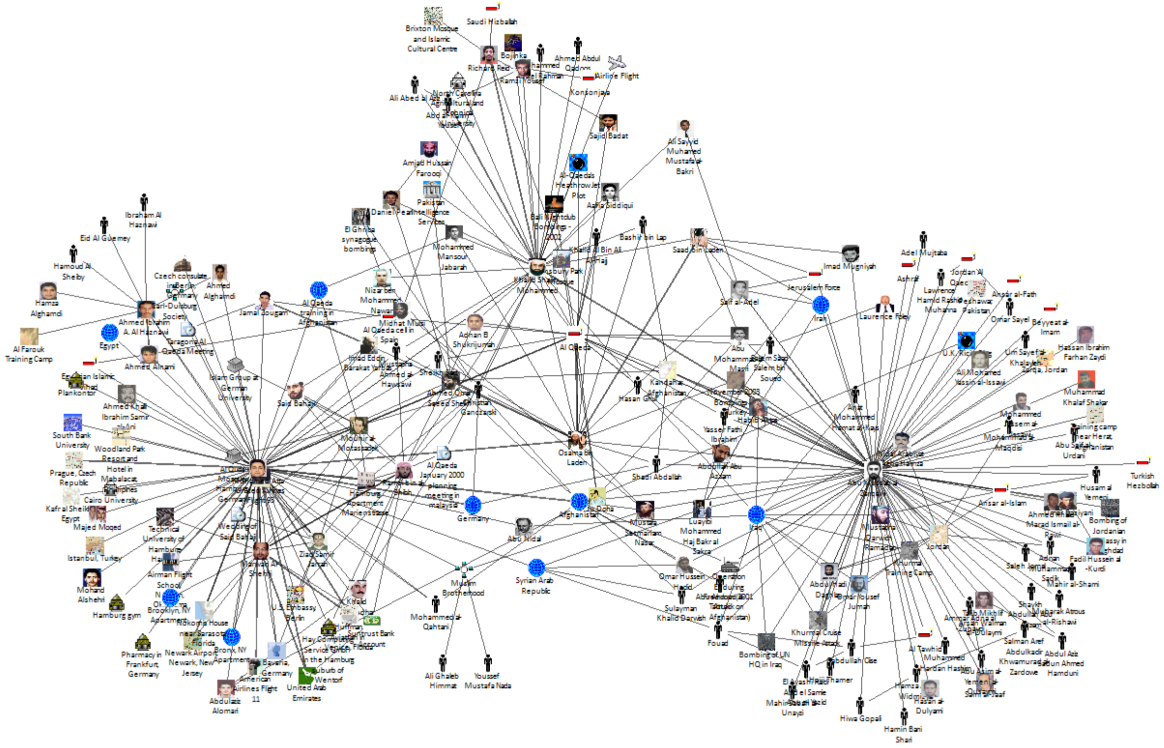
Example: News



Example: Books recommendation



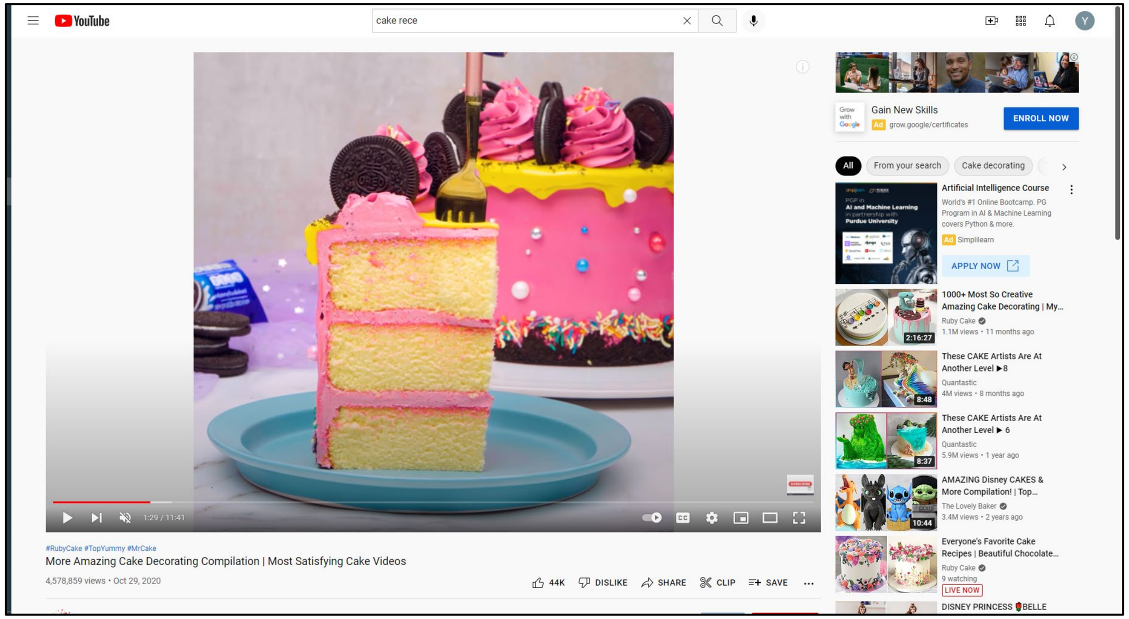
Example: Social networks



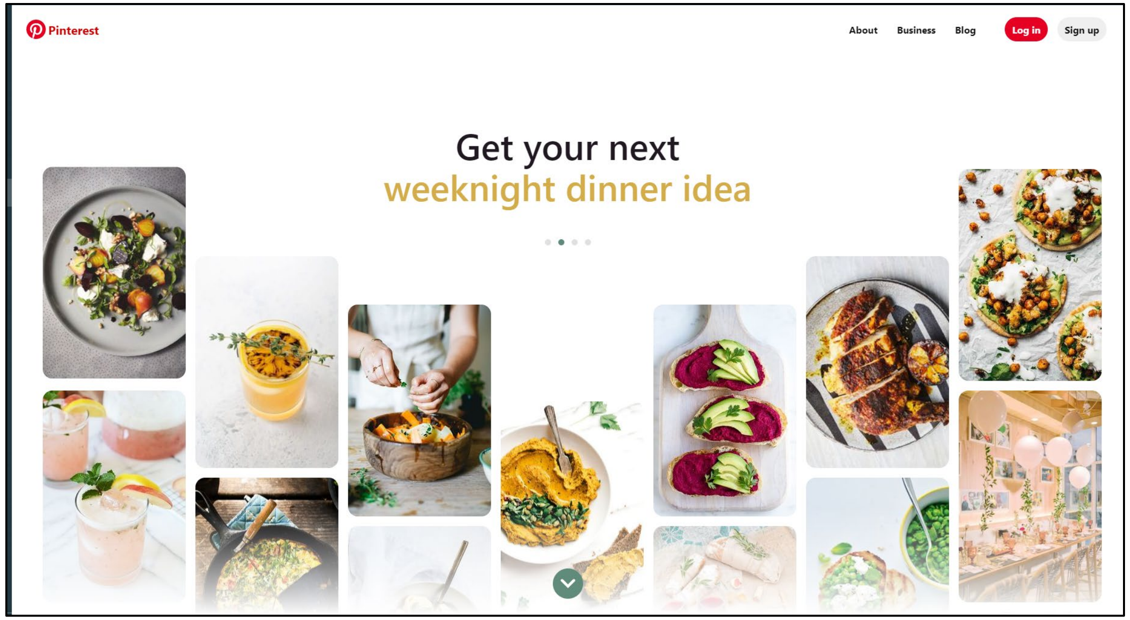
Images: https://www.vandelaydesign.com/free-social-media-icons/

https://fmsasg.com/socialnetworkanalysis/

Example: Image and video recommendations

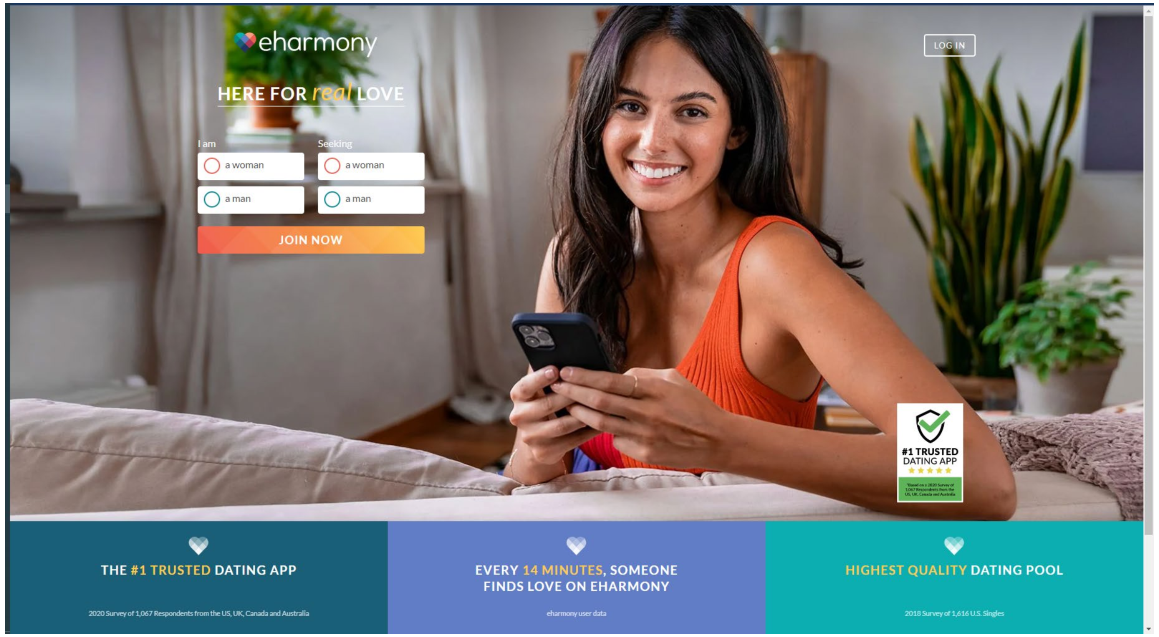


Youtube



Pinterest

Last example: Dating sites



- “Nearly 4% of all marriages in the US in 2012 were a result of eHarmony”. The analytics edge, Bertsimas et al.

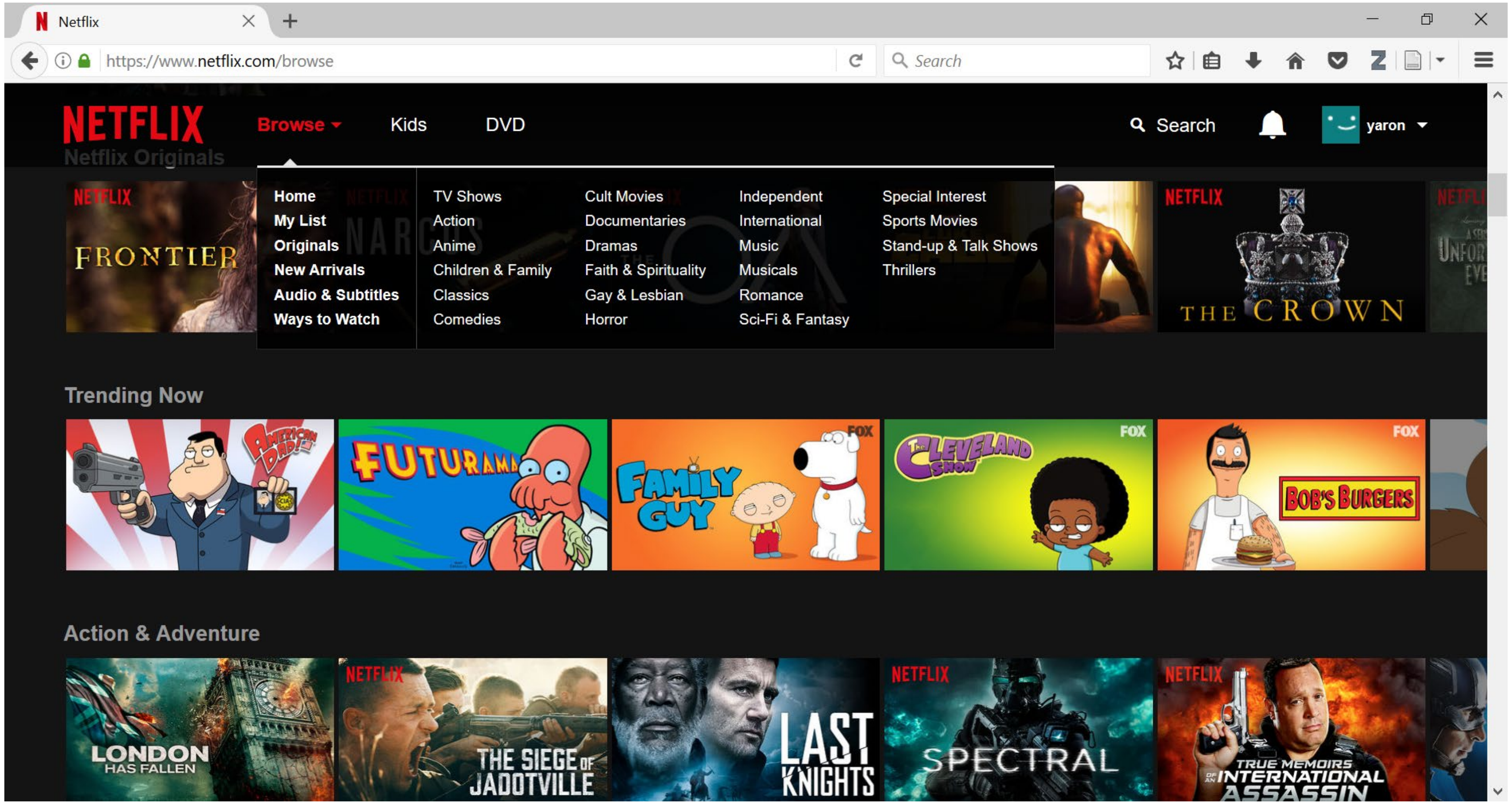
Why are they interesting and important?

- Important
  - Widely used predictive models
    - Entertainment – Hulo, Netflix, Pandora, Spotify, YouTube, …
    - Content – news websites, e-learning (coursera), …
    - E-commerce – online retailing (Amazon, Wallmart), …
    - Services – travel (Expedia), expert consultations (Yelp), …
    - Social media – Facebook, LinkedIn, Google+, …
  - Companies invest huge amount of resources in creating recommendation systems
  - Critical to increasing customer satisfaction and sales
- Interesting
  - A different class of predictive models
  - Rather than predicting on a single type of object characterized by features, here we often make prediction about the interaction between two types of objects that are characterized by different features

How do they work?

- Baseline predictors
- Content-based filtering
- Matrix factorization
- Graphical models
- Hybrid approach
- Collaborative filtering
- ANNs

Context: movie recommendation



Netflix

- 1997: founded, offers movie rentals by mail
- 1998: launched website
- 2000: Introduced the Cinematch recommendation system
  - Drives sales (60% of subscribers add recommended movies)
  - Keeps larger part of the library in circulation
  - 75% of prediction within 1/2 a star
  - Half of Netflix users give 5 star to recommended movies

Netflix – cont.

- 2006: launched the “Netflix prize”
  - Improve prediction accuracy by 10%
  - Prize: 1M dollar
  - 100M ratings, 480K users, 18K movies
  - Anonymized data (user IDs)
  - Significant impact on research
- 2007: video streaming
- 2009: winner announced
  - Combination of techniques including nearest neighbors and matrix factorization
- 2021: Revenue 29.7B

Example: movie recommendation

- Historical data: Users, Movies, Ratings

	Forest Gump (F)	Godfather (G)	Inception (I)	Jaws (J)
Amy (A)	5		4	3
Bob (B)	3	5	2	5
Carl (C)		3	5	4
Dan (D)	4	5	4	
Eva (E)	4	4		3

- Objective: predict Eva’s rating for Inception
- Simplified setting
  - In reality: size, sparsity, frequent changes, information about users (location, age), information about movies (genre, year), other interactions (like, dislikes, views without ratings, etc.), user experience (interface), privacy (target), data (collection, cleaning, accessing), recommendation (diversity), evaluation (metrics, CV)
- Nonetheless, context to discuss key algorithmic ideas

Vocabulary and notation

- U – the collection of users
- I – the collection of items (movies)
- $\boldsymbol{R}$ – the rating matrix
  - $r_{u,i}$ - rating of user u for item i
  - $\boldsymbol{r}_u$ - all rating of user u (row)
  - $\bar{r}_u$ - average ratings of user u
  - $\boldsymbol{r}_i$ - all rating for item i (column)
  - $\bar{r}_i$ - average ratings of item i
  - μ –average of all ratings
- Prediction
  - $p_{u,i}$ - prediction for the rating of user u for item i

	Forest Gump (F)	Godfather (G)	Inception (I)	Jaws (J)
Amy (A)	5		4	3
Bob (B)	3	5	2	5
Carl (C)		3	5	4
Dan (D)	4	5	4	
Eva (E)	4	4		3

Baseline predictors

- Simple methods that average predictions
- Useful for
  - Benchmarking against personalized methods
  - Building blocks in other methods
  - Predictions for new users
- Notation: $b_{u,i}$ - baseline prediction
- Global average $b_{u,i} = \mu$
- User average $b_{u,i} = \bar{r}_u$
- Item average $b_{u,i} = \bar{r}_i$

	Forest Gump (F)	Godfather (G)	Inception (I)	Jaws (J)	avg.
Amy (A)	5		4	3	**4.00**
Bob (B)	3	5	2	5	**3.75**
Carl (C)		3	5	4	**4.00**
Dan (D)	4	5	4		**4.33**
Eva (E)	4	4		3	**3.67**
avg.	**4**	**4.25**	**3.75**	**3.75**	**3.94**

Baseline predictors – cont.

- A combined user-item baseline predictor $$b_{u,i} = \mu + b_u + b_i$$

  where
  $$b_u = \frac{1}{|I_u|} \sum_{i \in I_u} (r_{u,i} - \mu)$$
  $$b_i = \frac{1}{|U_i|} \sum_{u \in U_i} (r_{u,i} - b_u - \mu)$$

- Regularization:
  $$b_u = \frac{1}{|I_u| + \beta_\mathrm{u}} \sum_{i \in I_u} (r_{u,i} - \mu)$$
  $$b_i = \frac{1}{|U_i| + \beta_\mathrm{i}} \sum_{u \in U_i} (r_{u,i} - b_u - \mu)$$

- Some methods use $b_{u,i}$ to normalize $\boldsymbol{R}$ by subtracting it from $\boldsymbol{R}$
  - Missing values can be set to zero
  - $\boldsymbol{R}$ can be stored more efficiently

Collaborative filtering

- Collaborative filtering (CF)
  - Very popular and effective approach
  - Key idea: predict based on other users
    - Similar users – predict similarly
    - Different users – predict differently
- Many variation of this basic idea

User-user collaborative filtering

- User-user CF
  - Finds similar user who watched a movie and use their predictions for a user who hasn't watched it
  - Requires defining a similarity function $s: U \times U \to \mathbb{R}$
  - Using s finds the K nearest neighbors $N \subseteq U$
- Prediction

$$p_{u,i} = \bar{r}_u + \frac{\sum_{u' \in N} s(u, u')(r_{u',i} - \bar{r}_{u'})}{\sum_{u' \in N} |s(u, u')|}$$

Examples of user similarity functions

- Correlation

$$s(u,v) = \frac{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_u \cap I_v} (r_{v,i} - \bar{r}_v)^2}}$$

- Cosine similarity

$$s(u,v) = \frac{\mathbf{r}_u \cdot \mathbf{r}_v}{\|\mathbf{r}_u\|_2 \|\mathbf{r}_v\|_2} = \frac{\sum_i r_{u,i} r_{v,i}}{\sqrt{\sum_i r_{u,i}^2} \sqrt{\sum_i r_{v,i}^2}}$$

User-user collaborative filtering - example

- Example
  - Correlation
  - K=2
- Goal: predict $p_{C,e}$
- Assume $s(C,A) = 0.832$ and $s(C,D) = -0.515$

	Batman Begins	*Alice in Wonderland*	*Dumb and Dumber*	*Equilibrium*
User A	4	?	3	5
User B	?	5	4	?
User C	5	4	2	?
User D	2	4	?	3
User E	3	4	5	?

$$p_{C,e} = \bar{r}_C + \frac{s(C,A)(r_{A,e} - \bar{r}_A) + s(C,D)(r_{D,e} - \bar{r}_D)}{|s(C,A)| + |s(C,D)|}$$

$$p_{C,e} = \frac{5+4+2}{3} + \frac{0.832\left(5 - \frac{4+3+5}{3}\right) - 0.515\left(3 - \frac{2+4+3}{3}\right)}{0.832 + 0.515}$$

User-user collaborative filtering – scalability

- The approach relies on quickly identifying nearest neighbors and computing similarity scores
- What happens if a user rates multiple movies?
- Alternative: use similarities between items

Item-item collaborative filtering

- Generate prediction using the user's own ratings for other items combined with those items' similarities to the target item
- Item similarity function: $s \colon I \times I \to \mathbb{R}$
  - Normalization helps (some movies are more popular)
  - Can be precomputed

	Batman Begins	*Alice in Wonderland*	*Dumb and Dumber*	*Equilibrium*
User A	4	?	3	5
User B	?	5	4	?
User C	5	4	2	?
User D	2	4	?	3
User E	3	4	5	?

- Prediction: $p_{u,i} = \frac{\sum_{j \in S} s(i,j) r_{u,j}}{\sum_{j \in S} |s(i,j)|}$

Item-item collaborative filtering - example

- Assume that
  - The 2 nearest neighbors of item e (Equilibrium) are items b (Batman begins) and d (Dumb and dumber)
  - $s(b,e) = 0.607$ and $s(d,e) = 0.382$

	Batman Begins	*Alice in Wonderland*	*Dumb and Dumber*	*Equilibrium*
User A	4	?	3	5
User B	?	5	4	?
User C	5	4	2	?
User D	2	4	?	3
User E	3	4	5	?

$$
\begin{aligned}
p_{C,e} &= \frac{s(b,e)\cdot r_{C,b} + s(d,e)\cdot r_{C,d}}{|s(b,e)| + |s(d,e)|} \\
&= \frac{0.607 * 5 + 0.382 * 2}{0.607 + 0.382} \\
&= 3.84
\end{aligned}
$$

Item-item collaborative filtering – “Cold start”

- What if I never rated movies before?

Matrix factorization

- Suppose we could represent each item by a short vector of numbers
  - E.g., for movies, the amount of violence (1-5), drama or comedy (0-1, 1 comedy, 0 drama), popularity of staff (1-10)
  - The king's speech: (1,0.1,8)
  - Pulp fiction: (5,0.5,9)
- Suppose we could represent users by equal weight preferences
  - (0.9,0.1,0.75) – user A likes violence, dramas, and famous casts
  - (0.2,0.9.0.2) – user B likes comedies, and doesn't care about the cast
- We could use it for prediction:
  - $(0.9,0.1,0.7) \cdot (1,0.1,8) = 6.91$ for user A and the King's speech

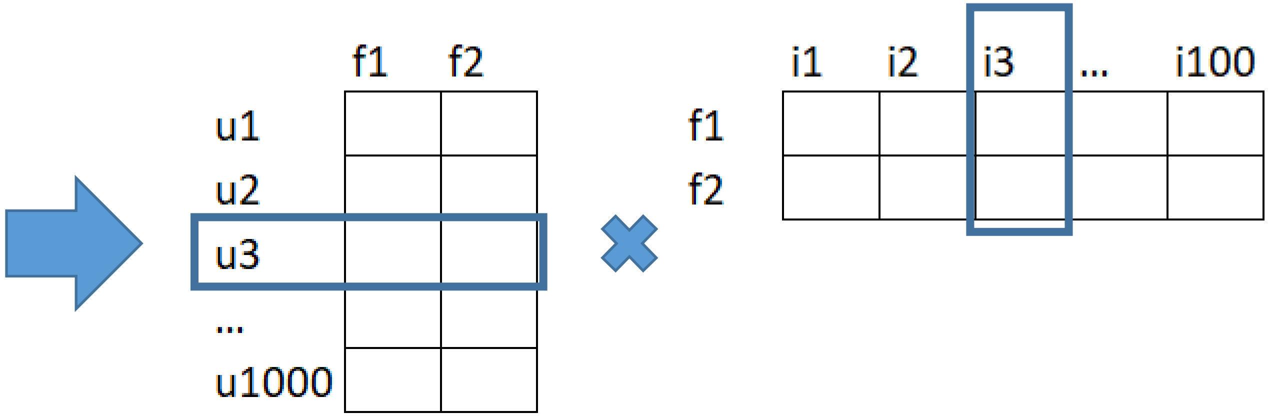
Matrix factorization – cont.

- How to find such vectors?
  - Our starting point is the rating matrix
  - Suppose in our data user $u^{(1)}$ rated movie A as 5 movie B as 3
  - User $u^{(1)}$: $(u_1^{(1)}, u_2^{(1)}, u_3^{(1)})$
  - Movie A: (a_1, a_2, a_3)
  - Movie B: (b_1, b_2, b_3)
  - We have
    - $u_1^{(1)} \cdot a_1 + u_2^{(1)} \cdot a_2 + u_3^{(1)} \cdot a_3 = 5$
    - $u_1^{(1)} \cdot b_1 + u_2^{(1)} \cdot b_2 + u_3^{(1)} \cdot b_3 = 3$
    - …
  - Solve the system of equations to find the variables

Matrix factorization – cont.

- Alternative view

	i1	i2	i3	...	i100
u1	3		4	2	3
u2			3		2
u3	4	2	5	5	
...		1		5	4
u1000	1	5	3		3



- Popular approach
- Returns features! (visualization, similarity between users and/or items); might be interpreted manually
- SVD (singular value decomposition) – an algorithm that can be used for matrix factorization
- Alternatively, SGD (stochastic gradient descent)
- Not easy to include other data

Probabilistic models

- Assumptions
  - There are K_m movie types (action, romantic comedy, etc.)
  - Each movie is a distribution over the types
  - Similarly, assume K_u user types, and each user is represented by a distribution over the user types
  - For each user type and movie type, there is a distribution over the ratings
- For specific user and movie we can predict
  - The probability that the user's rating is r : $\sum_{u \in 1 \ldots K_u} \sum_{i \in 1 \ldots K_m} P(u) P(i) P(r|u,i)$
  - Expected rating: $\sum_{u \in 1 \ldots K_u} \sum_{i \in 1 \ldots K_m} \sum_{r=1 \ldots 5} r \cdot P(u) P(i) P(r|u,i)$
- How to estimate probabilities?
  - Maximize loglikelihood
  - The EM algorithm

Probabilistic models – cont.

- Some similarities to matrix factorization

Data

	i1	i2	i3	…	i100
u1	2		3	5	5
u2			5		4
u3	4	2	4	3	
…		4		1	3
u1000	1	4	2		3

User types

	1	2	3
u1	0.2	0.7	0.1
u2	0.8	0.1	0.1
u3	…	…	…
…	…	…	…
u1000	…	…	…

Movie types

	i1	i2	i3	…	i100
1	0.8	0.7	…	…	…
2	0.2	0.3	…	…	…

Ratings

		Rating				
User type	Movie type	1	2	3	4	5
1	1	0.1	0.1	0.2	0.6	0.0
2	2	…	…	…	…	…
3	1	…	…	…	…	…
1	2	…	…	…	…	…
2	1	…	…	…	…	…
3	2	…	…	…	…	…

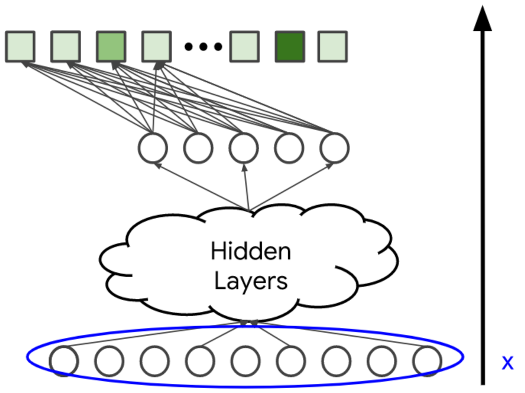
- Richer predictions (probabilities)
- Can be customized

Content-based filtering

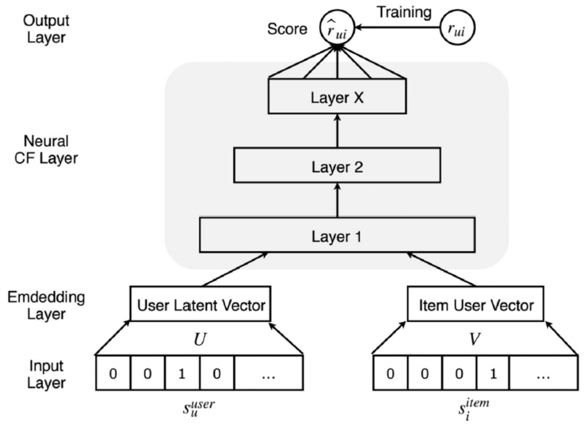
- Use only information about the user to make predictions
- Manually develop a similarity function between items (e.g., based on genre, year, director, etc.)
- Use weighted predictions
- Pros
  - Fast (use only data about the user)
  - Specializes
- Cons
  - Might over-specialize
  - Requires domain knowledge

Artificial neural networks

- Becoming increasingly more popular
- Example
  - Can use user data as features (information about the user, location, time, etc.)
  - Can use items as the output layer (probability for selecting an item)
- Computationally, more demanding
- Naturally support various features (including unstructured data)



Source: https://developers.google.com/machine-learning/recommendation/dnn/softmax



Source: 2019 Deep Learning Based Recommender System A Survey

Hybrid approaches

- General idea: combine multiple methods together
- Examples
  - Based on amount of data about a user apply collaborative or content-based filtering
  - Apply different methods to different subset of users (clusters, e.g., by country, age group, etc.)
- The winners of the Netflix challenge used a hybrid approach

surprise

- http://surpriselib.com/
- Python package for developing recommendation systems
- Contains implementation of common algorithms

- Keras: https://keras.io/examples/structured_data/collaborative_filtering_movielens/#create-the-model

Summary

- Recommendation is an important application of predictive modeling
  - Most businesses/organizations have multiple products, services, or styles that they need to recommend/personalize to users
- “Matching” intuition

References

- 2016 “**The analytics edge**” by Bertsimas, O`Hair, and Pulleyblank, Chapter 13
- 2009 **The BellKor Solution to the Netflix Grand Prize**, Koren
- 2010 **Collaborative Filtering Recommender Systems**, MD Ekstrand, JT Riedl, JA Konstan
- 2015 **Recommender system application developments: A survey**, Lu, Wu, Mao, Wang, Zhang
- 2019 **Deep Learning Based Recommender System: A Survey**, Zhang, Yao, Sun, Tay
- Lectures notes by Prof. Tommi Jaakola (6.867 Machine learning), available online
- Online tutorial on recommendation systems: https://developers.google.com/machine-learning/recommendation