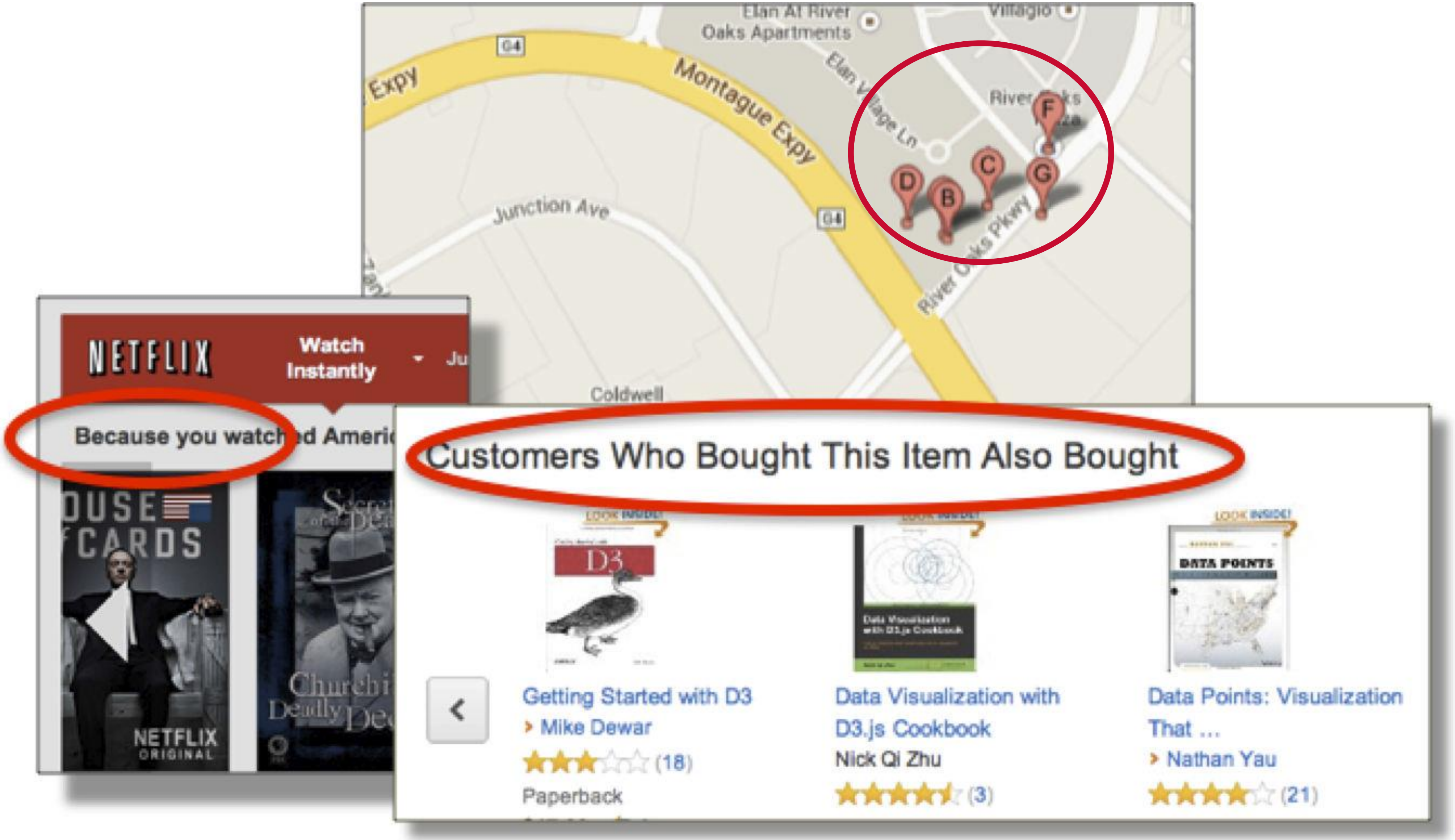
Recommendation Engine / Alternative Least Squares

MapR PS

Contents

- Recommendation Engine
- Apriori / K-NN
- Similarity Function
- Collaborative Filtering
- User-based, Item-based
- Content-based, Matrix Factorization methods
- Alternating Least Squares

Recommender examples in the wild



What is recommendation?

- **Recommendation** is a class of ML that seeks to predict a user's *preference* for or *rating* of an item

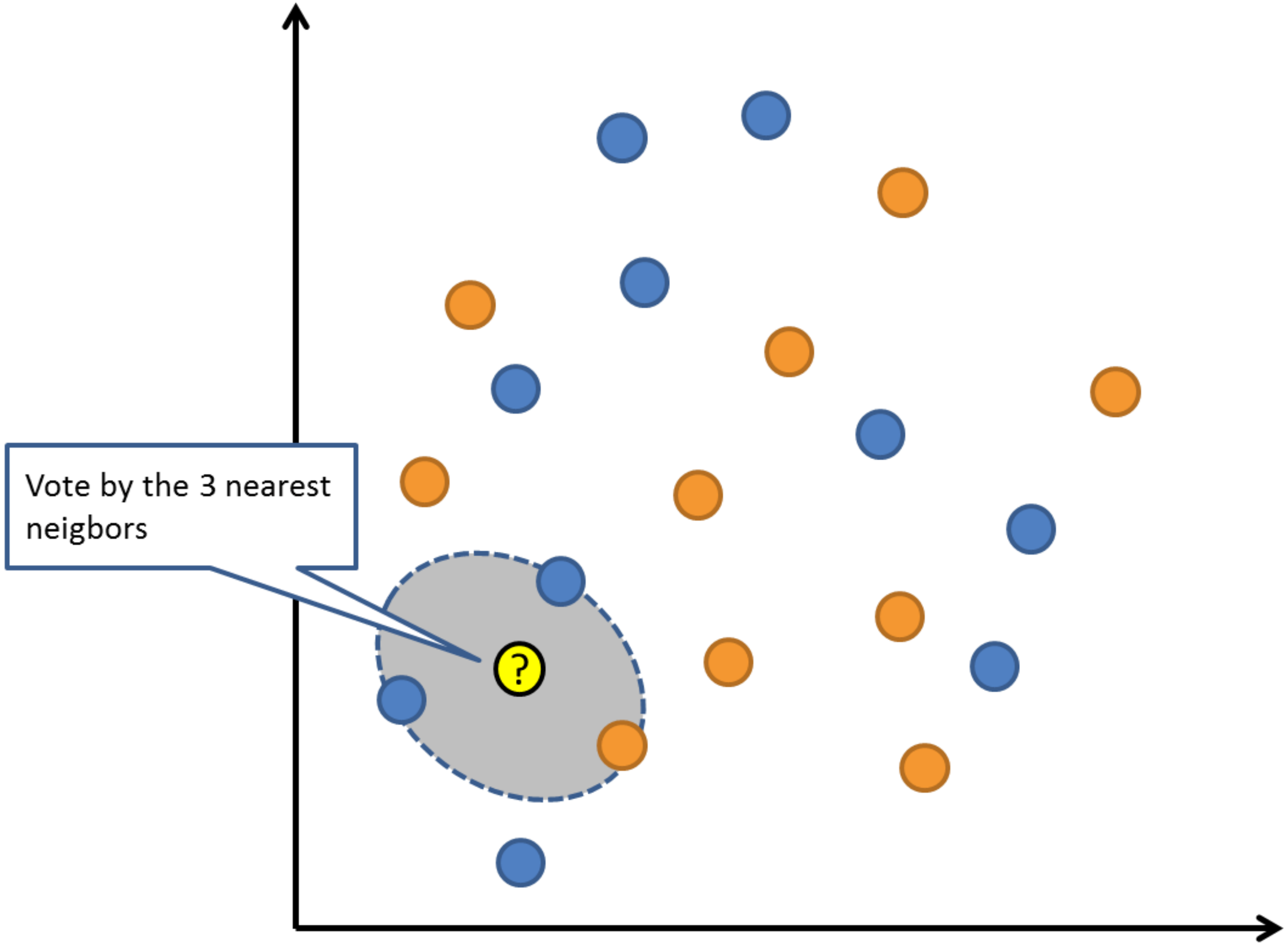
- **Recommender systems are used in industry to recommend:**
  - Books and other products (e.g Amazon)
  - Music (e.g. Pandora)
  - Movies (e.g. Netflix)
  - Restaurants (e.g. Yelp)
  - Jobs (e.g. LinkedIn)
  - . . . LOTS . . .

- **Main approaches to recommendation**
  - Collaborative filtering
  - Content-based filtering

Apriori Algorithm

$$
\begin{aligned}
&\text{Apriori}(T, \epsilon)\\
&\quad L_1 \leftarrow \{\text{large } 1-\text{itemsets}\}\\
&\quad k \leftarrow 2\\
&\quad \textbf{while } L_{k-1} \neq \emptyset\\
&\qquad C_k \leftarrow \{a \cup \{b\} \mid a \in L_{k-1} \wedge b \notin a\} - \{c \mid \{s \mid s \subseteq c \wedge |s| = k-1\} \nsubseteq L_{k-1}\}\\
&\qquad \textbf{for } \text{transactions } t \in T\\
&\qquad\quad C_t \leftarrow \{c \mid c \in C_k \wedge c \subseteq t\}\\
&\qquad\quad \textbf{for } \text{candidates } c \in C_t\\
&\qquad\qquad count[c] \leftarrow count[c] + 1\\
&\qquad L_k \leftarrow \{c \mid c \in C_k \wedge \ count[c] \geq \epsilon\}\\
&\qquad k \leftarrow k + 1\\
&\quad \textbf{return } \bigcup_k L_k
\end{aligned}
$$

K-Nearest Neighbor (KNN)



KNN algorithm

- **Find similar entities**
  - Use Similarity Function
- **Majority voting**
  - Problematic if class distribution is skewed
  - Could use distance as weight to compensate
  - Pick odd number for k to avoid ties
- **One of the oldest machine learning algorithms**
- **Works well in many cases**
- **How to decide similarity?**

Similarity metrics

- **Pearson correlation**
  - Ratio of covariance to product of standard deviatio
  - -1 ➔ inversely proportional; 0 ➔ no correlation; 1 ➔ directly proportional

$$w(a,i) = \frac{\sum_j (v_{a,j} - \overline{v}_a)(v_{i,j} - \overline{v}_i)}{\sqrt{\sum_j (v_{a,j} - \overline{v}_a)^2 \sum_j (v_{i,j} - \overline{v}_i)^2}}$$

- **Euclidean distance**
  - Coordinates represent item preferences (i1, i2, i3, ..
  - Smaller distance ➔ more similar (so return 1/(1+d))

$$w(a,i) = \sum_j \frac{v_{a,j}}{\sqrt{\sum_{k \in I_a} v_{a,k}^2}} \frac{v_{i,j}}{\sqrt{\sum_{k \in I_i} v_{i,k}^2}}$$

- **Tanimoto coefficient**
  - Ratio of intersection to union
  - Between 0 and 1 ➔ bigger is more similar

$$T_s(X,Y) = \frac{\sum_i (X_i \land Y_i)}{\sum_i (X_i \lor Y_i)}$$

Collaborative Filtering

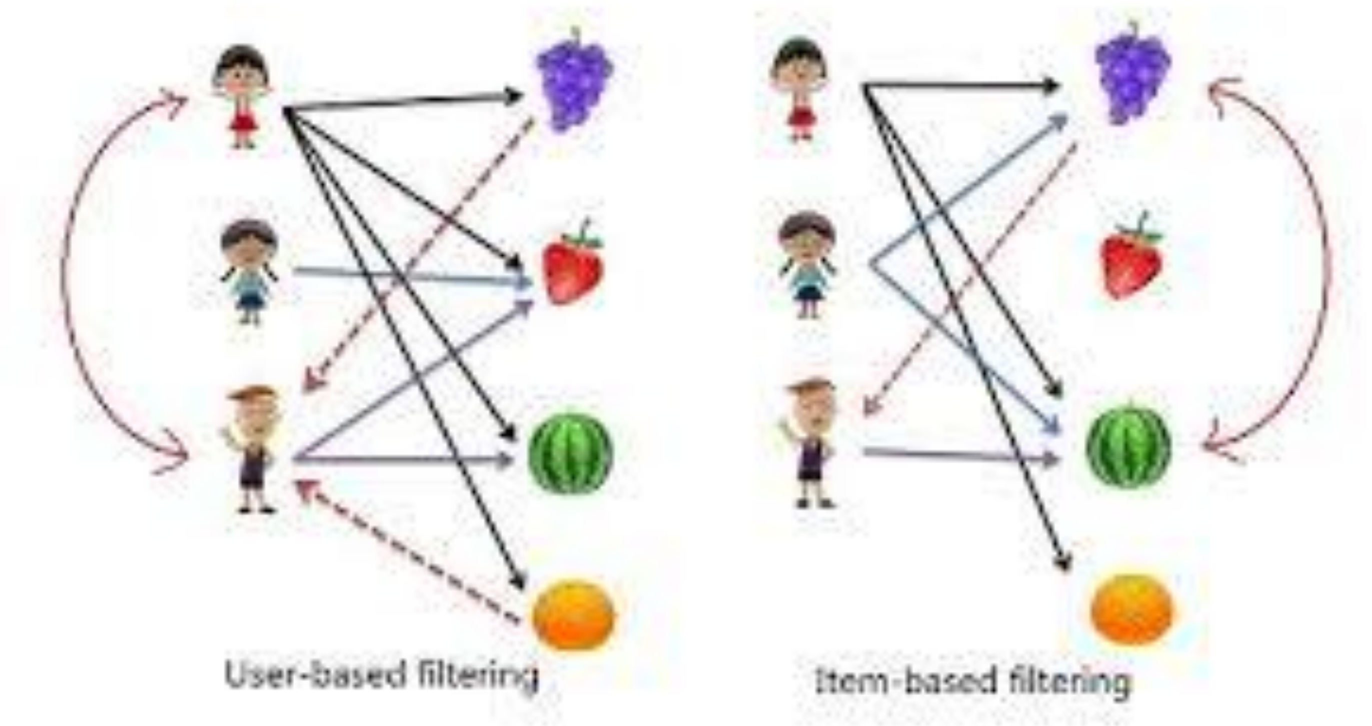
- Users = u1, u2, u3, u4 …
- Items = i1, i2, i3, i4 …
- Rating or purchase history = v11, v12, v13, v14, …
  - v12 means
  - User u1 rated Item i2 as v12
- How to would u1 rate i4 if u1 hasn't seen it yet?

$$\overline{v}_i = \frac{1}{|I_i|} \sum_{j \in I_i} v_{i,j}$$

$$w(a,i) = \frac{\sum_j (v_{a,j} - \overline{v}_a)(v_{i,j} - \overline{v}_i)}{\sqrt{\sum_j (v_{a,j} - \overline{v}_a)^2 \sum_j (v_{i,j} - \overline{v}_i)^2}}$$

$$p_{a,j} = \overline{v}_a + \kappa \sum_{i=1}^{n} w(a,i)(v_{i,j} - \overline{v}_i)$$

User-based vs item-based filtering



What to tune for better performance?

- **Model type**
  - User-based
  - Item-based
  - Content-based

- **Distance metric**
  - Euclidean
  - Tanimoto
  - Loglikelihood

- **Neighborhood size**
  - 2
  - 3
  - 200

Matrix Factorization

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} | & | \\ u_1 & u_2 \\ | & | \end{bmatrix} \times \begin{bmatrix} \lambda_1 & \varnothing \\ \varnothing & \lambda_2 \end{bmatrix} \times \begin{bmatrix} \text{——} & v_1 & \text{——} \\ \text{——} & v_2 & \text{——} \end{bmatrix}$$

Matrix Factorization

- **A = U Λ V**^{T} - example:

variance ('spread') on the v1 axis

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \\ 0 & 0.53 \\ 0 & 0.80 \\ 0 & 0.27 \end{bmatrix} \times \begin{bmatrix} 9.64 & 0 \\ 0 & 5.29 \end{bmatrix} \times \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix}$$

Matrix Factorization Based Recommendation

- **SVD:** $M_k = U_k \times \Sigma_k \times V_k^T$

U_k	Dim1	Dim2
Alice	0.47	-0.30
Bob	-0.44	0.23
Mary	0.70	-0.06
Sue	0.31	0.93

V_k^T	Terminator	Die Hard	Twins	Eat Pray Love	Pretty Woman
Dim1	-0.44	-0.57	0.06	0.38	0.57
Dim2	0.58	-0.66	0.26	0.18	-0.36

Σ_k	Dim1	Dim2
Dim1	5.63	0
Dim2	0	3.23

- **Prediction:** $\hat{r}_{ui} = \bar{r}_u + U_k(Alice) \times \Sigma_k \times V_k^T(EPL)$

= 3 + 0.84 = 3.84

Matrix Factorization Based Recommendation, ALS

- $\tilde{v}_{ij} = u_i^k \times i_j^k$

			Eat Pray Love
Alice	0.47	-0.30	0.38
			0.18

- $e = v_{ij} - \tilde{v}_{ij}$
- $u_i^k = u_i^k + \gamma(e \cdot i_j^k - \delta \cdot u_i^k)$
- $i_j^k = i_j^k + \gamma(e \cdot u_i^k - \delta \cdot i_j^k)$

ALS

- Inspired by Matrix Factorization
- Has almost nothing to do with Matrix Factorization
- Rather similar to Neural Network
- Can be quite unstable
- Quite random
- Just works fine for some cases

Challenges with Recommendation Engines

Challenge	Description
Cold start	No user history means no associations day 1
Scale	Huge number of products and users requires a lot of computation
Sparsity	Most users express very little behavior with very few items and no behavior on the vast majority of items

- What are the two main algorithms for recommendation engine?

- What is the fundamental problems in using collaborative filtering?

- What is the base idea of collaborative filtering?

- What is the relation between Matrix Factorization and ALS?