LESSON 12 : Recommender Systems

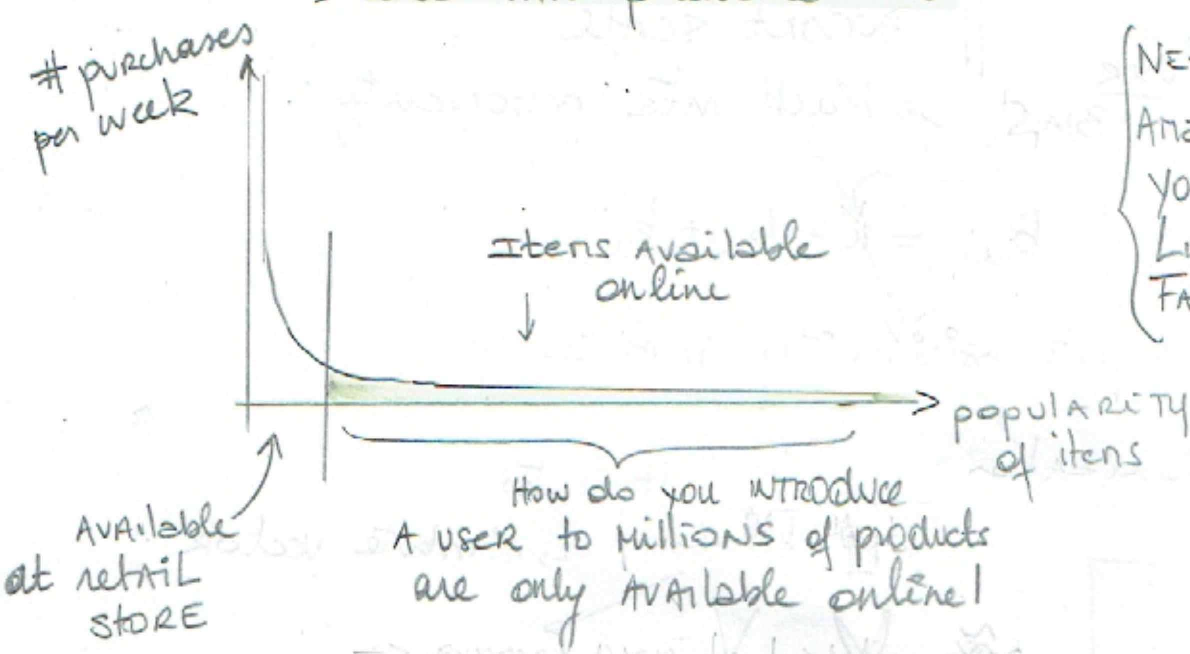
→ popularity ~ Netflix challenge in 2005 → 1M$ I

Predict a user's interest in a product or service

USER :
Receive relevant recommendations where there are many choices
ERA of scarcity to an era of abundance
↳ internet provides shelf WITHOUT cost!

VENDOR
- upselling opportunity
- loyalty ↑

→ LONG TAIL phenomenon



- Netflix
- Amazon
- Youtube
- LinkedIn
- Facebook

Type of recommendation

(1) hand created → Youtube Trending

Focus → (2) Tailored to individual users!

Problem statement : Netflix

4 pax ↓ ; ← 5 MOVIES

	1	2	3	4	5
P_1	1			3	
P_2		5			
P_3				1	
P_4	2				4

↳ RATINGS : 1 → 5

← USER/ITEM MATRIX
→ SPARSE (as high as 95% sparsity)
→ GOAL is to fill in the gaps!

How to gather Ratings?

- ① Ask! → Explicit
- ② implicit info: profile info, click info, facebook, purchase history, cursor movements, ..
  - → only relevant to high rating items

Most companies use both.

How to find Rating matrix?

① Content-based

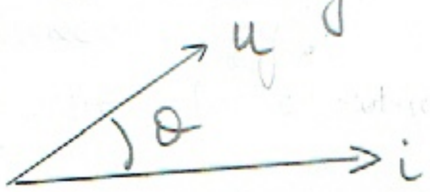
recommend items based on what the user has rated highly before! → user profile 'vector' → (thriller, yes) (genre, oscar)

→ item needs a profile (vector)
  ex: movie (genre, oscar, director, ...)

profile parameters need to be the same

→ match item profile to profile user
  - item profile: how much item i satisfies property k.
  - profile user: ↳ developed by explicit or implicit data → how much user values property k

↓ calculate 'similarity'



→ dot product

$$\bar{u} \cdot \bar{i} = |u| \cdot |i| \cos \theta$$

Assume: ANA
- likes thrillers → 5
- likes movies that won Oscar → 4
- likes Robert de Niro → 5

$$\begin{pmatrix} 5 \\ 4 \\ 5 \end{pmatrix}$$

person C

what is Rating for JAWS (movie 3), Cape Fear (movie 5) movie 3 : Jaws

5 : Cape Fear

Jaws profile $\begin{pmatrix} 5 \\ 0 \\ 0 \end{pmatrix}$ Cape fear profile $\begin{pmatrix} 5 \\ 0 \\ 5 \end{pmatrix}$

Jaws $\rightarrow$ $\begin{pmatrix} 5 \\ 4 \\ 5 \end{pmatrix} \cdot \begin{pmatrix} 5 \\ 0 \\ 0 \end{pmatrix} = \frac{25}{15} = 1.3$ $\begin{pmatrix} 5 \\ 4 \\ 5 \end{pmatrix} \begin{pmatrix} 5 \\ 0 \\ 5 \end{pmatrix} = \frac{50}{15} = 3.3$

basis for Ranking

Conclusion : * you can start on day 1 $\rightarrow$ good for cold start!

\+ * item profile does not depend on other users
no first rater problem

* more logical to understand where the recommendation comes from

\- * finding features is difficult and sometimes subjective

* overspecialization $\rightarrow$ no risk

* Not able to exploit judgement from other users

* cold start $\rightarrow$ no user profile

$\hookrightarrow$ Ask user to complete a form
with hints
assigne general profile

2 Collaborative filtering

↳ relies on (user, item) interactions

purchases
likes

↳ find 'similar' users

1) memory-based → neighbor-based

user A ~ user B
similar

user A → movie 1 R: 4, movie 2 R: 4

user B → movie 1 R: 4

→ recommend movie 2 to user B

user A → movie 1 R: 4, movie 2 R: 5

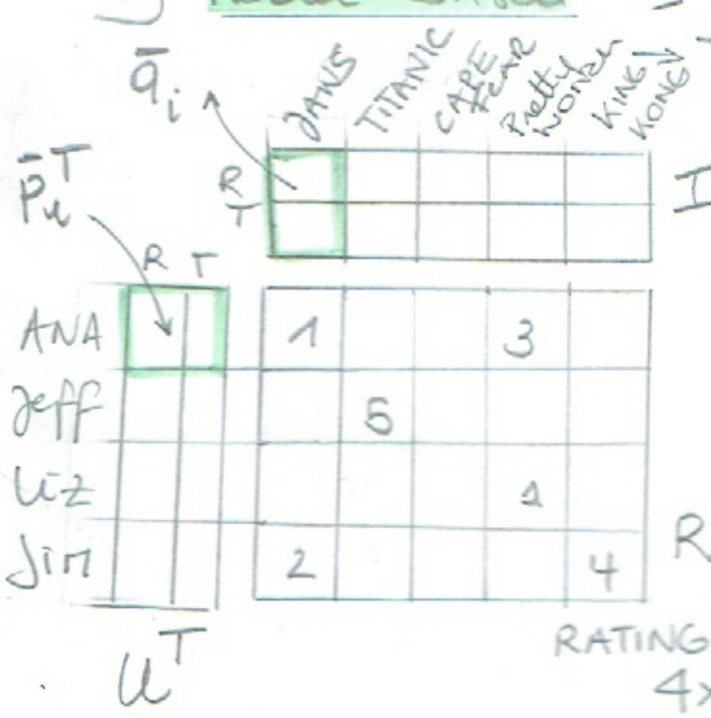
movie 3

and movie 3 ~ movie 2
movie 1

→ recommend movie3 to user A

2) Model-Based → Machine learning (supervised)

→ Train model on sparse entries



R: Romance
T: Thriller

RATING MATRIX
4x5

< sparse → sometimes up to 95%
< dependencies → low rank

next → **Factorization of Rating Matrix** (5)

↳ deconpose complex MATRIX IN (low Rank) Sinpler MATRICES

ex SVD (SINGULAR Value Deconp)

$A = U \Sigma V^T$

U: ROT

Σ: STRETCH

V^T: ROT

SVD not Applicable to sparse MATRICES

FIND LAtent Structure IN observed RATings

↓

highly correlated terms can be SUMMARIZED AS LAtent factors | → choose features!

Romance Thriller

features is hyper parameter

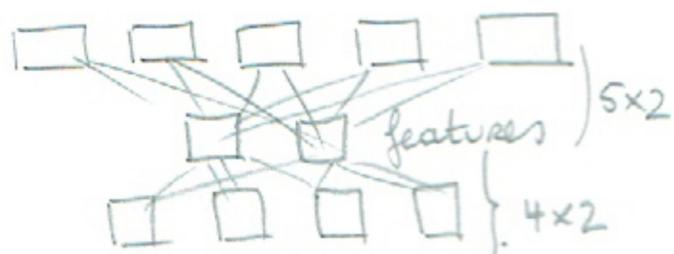
$$\boxed{R = U^T I}$$

U^T: User matrix

I: ↳ items MATRIX

features allow us to reduce need for memory

→ lower dinensional problem



18

vs 20

→ back to page 4 → Add features < R, T

Goal: find U^T and I so that MSE is minimal.

↳ | GRADIENT Descent
| Neural Networks

	R	T	[illegible]	TiT	CF	PW	KK
R			1				
T			4				
ANA	0.5	0.1	1			3	
Jeff				5			
Liz						1	
Jim.			2				4

$\bar{p}_u^T = \begin{pmatrix} 0.5 \\ 0.1 \end{pmatrix}$

$\bar{q}_i = \begin{pmatrix} 1 \\ 5 \end{pmatrix}$

initialize!

$\downarrow$

$\bar{p}_u^T \cdot \bar{q}_i = 0.5 + 0.4$
$= 0.9$

compare with entry
1.0

ERROR
$= 0.1$

update $\bar{p}_i^T, \bar{q}_i$ → find minimum using Gradient Descent

→ Do same with other sparse entries!

→ total 6 entries

→ this will give you U^T, I that minimizes loss on sparse entries!

→ once U^T, I are know → use U^T, I to fill in blanks using dot product!

Gradient Descent

$R = U^T I$

$$C = \frac{1}{2} \sum (\overbrace{r_{ui} - p_u^T q_i}^{e_{ui} = \text{ERROR}})^2$$

$$\frac{\partial C}{\partial p_u} \rightarrow (r_{ui} - p_u^T q_i)(-1) q_i$$

$$\frac{\partial C}{\partial q_i} \rightarrow (r_{ui} - p_u^T q_i)(-1) p_u$$

$$\rightarrow \frac{\partial C}{\partial p_u} = (\underbrace{-r_{ui} + p_u^T q_i}_{-e_{ui}}) q_i$$

$$\rightarrow \boxed{\begin{array}{l} p_u^T \leftarrow \bar{p}_u + \alpha e_{ui} q_i \\ q_i \leftarrow \bar{q}_i + \alpha e_{ui} p_u \end{array}}$$

update rules for p, q.

→ iterate till no significant changes → MINIMUM.

Neural Nets

↓

Several layers are crossed going from INPUT to OUTPUT

→ U^T, I are found via backpropagation!

Tensorflow

- 2015 google
- Large DATASETS
- Python / Keras
- multicore + distributed
- GPU TPU

REGULARIZATION

$$C = \frac{1}{2} \sum (r_{ui} - p_u^T q_i) + \underbrace{\lambda \| p_u \|^2 + \lambda \| q_i \|^2}_{L2 \text{ regularization}}$$

↓

2 UNKNOWN + NOT CONVEX !!

→ ALS (alternate least squares)

↳ keeps $\bar{p}_u$ constant and updates q_i

$\bar{q}_i$ $\bar{p}_u$

↓ for: Large Scale

SPARK

alternation makes C a quadratic for every iteration

→ CONVEX → GLOBAL MINIMUM!

ISSUES WITH CF

1. cold start
   ↳ solution
   - std profile
   - content-based at start
2. issue when users share ACCOUNT (Netflix)
3. USER bias
   - optimist
   - pessimist

Average rating