# recommendation engines

# stuff to learn today:

1. name different kinds of recommendation engines
2. explain the concepts behind each kind of engine
3. pros and cons of the different engines + when to use which

# **part 1**: types of recommendation engines, and **why**???

- three main types of recommendation engines:
  a. non-personalized
  b. content-based
  c. collaborative filtering

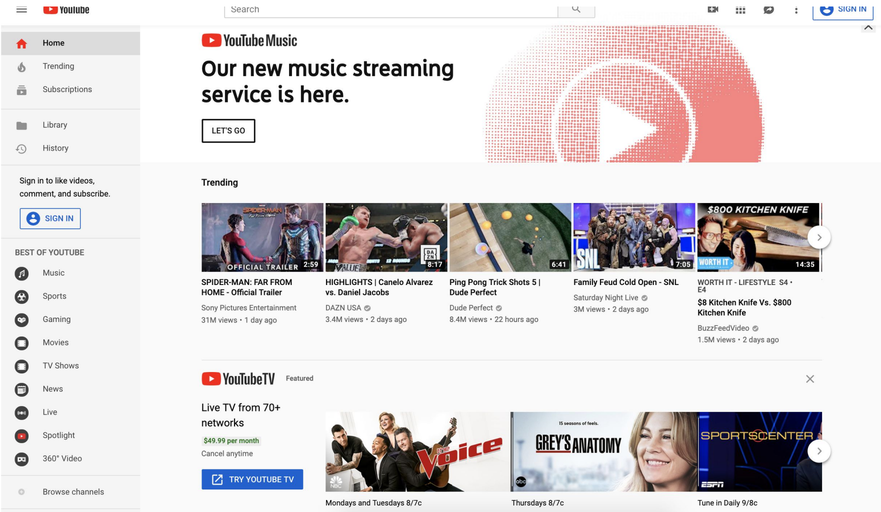
- why do we need recommendation engines + what are some examples?

# part 2: non-personalized recommendation engines

- top-rated items
- most bought/watched/consumed items
- items who will give companies highest ROI

- easy to make recommendations (same recommendations for everyone)

- is it the most effective?



# **part 3**: content-based recommendation engines

- makes recommendations based on an item's **features**
- models we already know that can do this!!

- how would you build a recommendation engine using models you already know?

| Items | | | | | | |
|---|---|---|---|---|---|---|
| Features | | **A** | **B** | **C** | **D** | **...** |
| | **Genre** | 2 | 3 | 5 | 1 | ... |
| | **Actor** | 5 | 4 | 2 | 1 | ... |
| | **Director** | 1 | 1 | 1 | 3 | ... |
| | **Year** | 3 | 4 | 5 | 2 | ... |
| | **IMDB Ratings** | 5 | 4 | 1 | 2 | ... |

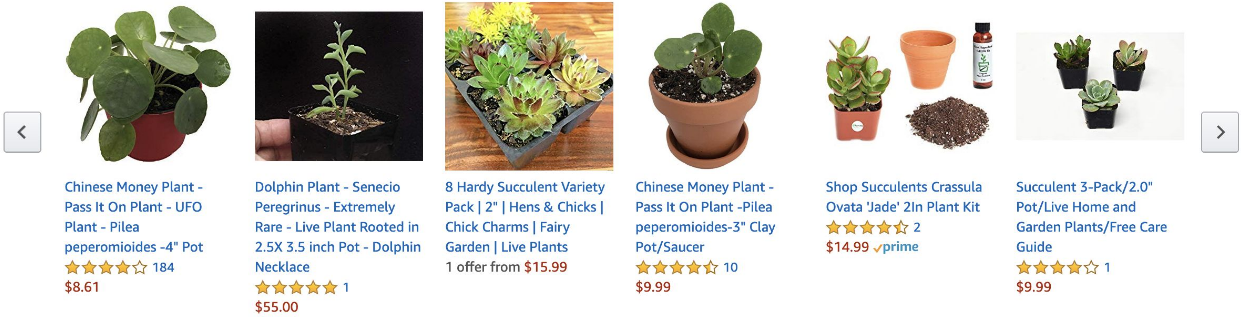
# similarity metrics

- euclidean distance
- jaccard index
- cosine similarity
- adjusted cosine similarity
- pearson correlation

| Items | | | | | | |
|---|---|---|---|---|---|---|
| | | A | B | C | D | ... |
| Features | Genre | 2 | 3 | 5 | 1 | ... |
| | Actor | 5 | 4 | 2 | 1 | ... |
| | Director | 1 | 1 | 1 | 3 | ... |
| | Year | 3 | 4 | 5 | 2 | ... |
| | IMDB Ratings | 5 | 4 | 1 | 2 | ... |

# **part 3**: content-based recommendation engines

- what are some pros and some pitfalls of content-based recommendations?



# part 4: collaborative filtering -- the utility matrix

- a utility matrix shows user ratings of different items (usually sparse 😞)

- the idea is to fill in the blanks, and recommend items with the highest predictions

| | Movie 1 | Movie 2 | Movie ... | Movie N |
|---|---|---|---|---|
| User 1 | 1 | BLANK | BLANK | 3 |
| User 2 | BLANK | 5 | BLANK | 3 |
| User 3 | BLANK | BLANK | 1 | BLANK |
| User 4 | 2 | 3 | BLANK | BLANK |
| User 5 | BLANK | BLANK | 1 | BLANK |
| User 6 | 4 | BLANK | 5 | BLANK |
| User 7 | BLANK | 4 | BLANK | BLANK |
| User ... | BLANK | 3 | BLANK | BLANK |
| User m | BLANK | BLANK | BLANK | 4 |

| | Movie 1 | Movie 2 | Movie ... | Movie N |
|---|---|---|---|---|
| User 1 | 1 | 4 | 2 | 3 |
| User 2 | 1 | 5 | 3 | 3 |
| User 3 | 2.5 | 2.8 | 1 | 3.5 |
| User 4 | 2 | 3 | 2 | 3.5 |
| User 5 | 2.5 | 2.8 | 1 | 3.1 |
| User 6 | 4 | 1.2 | 5 | 1.4 |
| User 7 | 1 | 4 | 2.5 | 3 |
| User ... | 2 | 3 | 2 | 3 |
| User m | 1 | 4 | 2 | 4 |

# **part 4**: collaborative filtering

- recommends items **based on ratings of other users**

- different ways to do collaborative filtering:

    - memory-based (aka neighborhood-based)
        - user-user similarity
        - item-item similarity

    - model-based (matrix factorization)

# part 4a: user-user collaborative filtering

filling up the utility matrix based on **user** similarities

to get recommendations for user X:

1. get user similarity values
2. the predicted rating for item A is a weighted average of others' ratings

| | U1 | U2 | U3 | U4 |
|---|---|---|---|---|
| I1 | 4 | 2 | 3 | 5 |
| I2 | 3 | 2 | 4 | 2 |
| I3 | ? | 4 | 5 | 4 |
| I4 | 3 | 2 | 4 | 4 |

| cos | U1 |
|---|---|
| U2 | 0.65 |
| U3 | 0.76 |
| U4 | 0.83 |

sum = 2.24

? = (0.65*4 + 0.76*5 + 0.83*4)/2.24
= **4.34**

# **part 4a**: collaborative filtering -- pros and cons

- personalized for each user

- computationally heavy
- popularity bias
- the **cold start** problem

# **part 4b**: item-item collaborative filtering

filling up the utility matrix based on **item** similarities

to get recommendations for user X:

1. get ITEM similarity values
2. the predicted rating for item A is a weighted average of other **items**

| | U1 | U2 | U3 | U4 |
|---|---|---|---|---|
| **I1** | 4 | 2 | 3 | 5 |
| **I2** | 3 | 2 | 4 | 2 |
| **I3** | ? | 4 | 5 | 4 |
| **I4** | 3 | 2 | 4 | 4 |

| cos | I3 |
|---|---|
| **I1** | 0.78 |
| **I2** | 0.83 |
| **I4** | 0.87 |

sum = 2.48

? = (0.78*4 + 0.83*3 + 0.87*3)/2.48
= **3.31**

# **part 4b**: user-user vs item-item

which is better?????

- in general, item-item has proven to be more effective
- it's hard to predict users' *unique* tastes

time complexity (for ***m*** users and ***n*** items)

- user-user: $O(m^2n)$
- item-item: $O(mn^2)$
- which would be faster if **m > n** (more users than items)?

similarity metrics:

- experimentally, **Pearson correlation** has demonstrated to be the best

# **part 4c**: model-based collaborative filtering

singular value decomposition:

- another way to fill in the utility matrix via matrix factorization

modified SVD in recommendation engines:

- breaks down the utility matrix into a **user matrix** and an **item matrix**
- the other dimensions are latent features
- gradient descent using Alternating Least Squares to preserve the relationship between items and between users (parallelizable)

very math, but best-in-class models use some form of SVD

# **part 4c**: modified SVD for filling in utility matrices

| | U1 | U2 | U3 | U4 |
|---|---|---|---|---|
| **I1** | 4 | 2 | 3 | 5 |
| **I2** | 3 | 2 | 4 | 2 |
| **I3** | ? | 4 | 5 | 4 |
| **I4** | 3 | 2 | 4 | 4 |

≈

| | | |
|---|---|---|
| **I1** | × | × |
| **I2** | × | × |
| **I3** | × | × |
| **I4** | × | × |

latent features

•

| U1 | U2 | U3 | U4 |
|---|---|---|---|
| × | × | × | × |
| × | × | × | × |

latent features

# **part 4c**: gradient descent with Alternating Least Squares

| | | |
|---|---|---|
| **I1** | 1 | 1 |
| **I2** | 1 | 1 |
| **I3** | 1 | 1 |
| **I4** | 1 | 1 |

•

| **U1** | **U2** | **U3** | **U4** |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |

=

| | **U1** | **U2** | **U3** | **U4** |
|---|---|---|---|---|
| **I1** | 2 | 2 | 2 | 2 |
| **I2** | 2 | 2 | 2 | 2 |
| **I3** | 2 | 2 | 2 | 2 |
| **I4** | 2 | 2 | 2 | 2 |

| OG | **U1** | **U2** | **U3** | **U4** |
|---|---|---|---|---|
| **I1** | 4 | 2 | 3 | 5 |
| **I2** | 3 | 2 | 4 | 2 |
| **I3** | ? | 4 | 5 | 4 |
| **I4** | 3 | 2 | 4 | 4 |

RMSE = 1.75

# **part 4c**: gradient descent with Alternating Least Squares

| | | |
|---|---|---|
| I1 | x | 1 |
| I2 | 1 | 1 |
| I3 | 1 | 1 |
| I4 | 1 | 1 |

•

| U1 | U2 | U3 | U4 |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |

=

| | U1 | U2 | U3 | U4 |
|---|---|---|---|---|
| I1 | x+1 | x+1 | x+1 | x+1 |
| I2 | 2 | 2 | 2 | 2 |
| I3 | 2 | 2 | 2 | 2 |
| I4 | 2 | 2 | 2 | 2 |

| OG | U1 | U2 | U3 | U4 |
|---|---|---|---|---|
| I1 | 4 | 2 | 3 | 5 |
| I2 | 3 | 2 | 4 | 2 |
| I3 | ? | 4 | 5 | 4 |
| I4 | 3 | 2 | 4 | 4 |

Finding **x** to minimize:

$(4-(x+1))^2 + (2-(x+1))^2 + (3-(x+1))^2 + (5-(x+1))^2$

setting $d/dx = 0$, $x =$ **2.5**

# **part 4c**: gradient descent with Alternating Least Squares

| | | |
|---|---|---|
| **I1** | 2.5 | 1 |
| **I2** | 1 | 1 |
| **I3** | 1 | 1 |
| **I4** | 1 | 1 |

•

| **U1** | **U2** | **U3** | **U4** |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |

=

| | **U1** | **U2** | **U3** | **U4** |
|---|---|---|---|---|
| **I1** | 3.5 | 3.5 | 3.5 | 3.5 |
| **I2** | 2 | 2 | 2 | 2 |
| **I3** | 2 | 2 | 2 | 2 |
| **I4** | 2 | 2 | 2 | 2 |

| OG | **U1** | **U2** | **U3** | **U4** |
|---|---|---|---|---|
| **I1** | 4 | 2 | 3 | 5 |
| **I2** | 3 | 2 | 4 | 2 |
| **I3** | ? | 4 | 5 | 4 |
| **I4** | 3 | 2 | 4 | 4 |

RMSE = 1.58!!

# stuff we've learned

recommendation engines!!!

<table>
<tr><td colspan="3">1. non-personalized</td></tr>
<tr><td colspan="3">2. content-based</td></tr>
<tr><td rowspan="3">collaborative filtering</td><td rowspan="2">memory-based</td><td>3. user-user</td></tr>
<tr><td>4. item-item</td></tr>
<tr><td>model-based</td><td>5. SVD</td></tr>
</table>