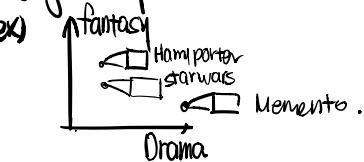


Building a simple Vector-based model

- content based filtering : uses item features to recommend new items that are similar to what the user has liked in the past.
- "learn How to Measure the similarity of elements in an embedding space"

Similarity Measures

- embedding : map from our collection of items to some finite dimensional vector space.



- Similarity measure : a metric for items in an embedding space

- * dot product.
- * cosine similarity

Building a user vector.

| | | Fantasy | Action | Cartoon | Drama | Comedy |
|-------------|------|---------|--------|---------|-------|--------|
| Star Wars | 1.0 | 0 | 0 | 1 | 0 | 1 |
| Incredibles | 1.0 | 0 | 1 | 0 | 0 | 0 |
| Toy Story | 1.0 | 0 | 0 | 0 | 0 | 1 |
| Star Trek | 0.71 | 1 | 0 | 0 | 0 | 1 |
| Avatar | 0.37 | 0 | 1 | 0 | 0 | 1 |
| | | 1 | 1 | 1 | 1 | 1 |
| | | 1 | 1 | 1 | 1 | 1 |

$$= \begin{pmatrix} 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 4 & 0 \\ 10 & 0 & 0 & 0 & 10 \end{pmatrix}$$

↓ Aggregate (Σ)

$$(10, 0, 1, 4, 11)$$

↓ Normalization

$$(0.26, 0, 0.18, 0.11, 0.45)$$

= User feature vector

* comedy (0.45) has the largest value.

| | | | | | |
|--|------|---|------|------|------|
| | 0.26 | 0 | 0.18 | 0.11 | 0.45 |
| | 1 | 1 | 0 | 0 | 0 |
| | 1 | 1 | 0 | 1 | 0 |
| | 0 | 1 | 1 | 0 | 1 |
| | 0 | 0 | 0 | 1 | 0 |

$$\Sigma = \begin{pmatrix} 0.26 & 0 & 0 & 0 & 0.45 \\ 0.26 & 0 & 0 & 0.11 & 0 \\ 0 & 0 & 0.18 & 0 & 0.45 \\ 0 & 0 & 0 & 0.11 & 0 \end{pmatrix}$$


$$\Sigma = \begin{pmatrix} 0.71 \\ 0.37 \\ 0.63 \\ 0.11 \end{pmatrix}$$

User movie rating

Making Recommendation for many users.

| | Action | Sci-Fi | Comedy | Cartoon | Drama |
|--------|--------|--------|--------|---------|-------|
| User 1 | 4 | 6 | 8 | | |
| User 2 | | | 10 | 8 | |
| User 3 | | 6 | | | 3 |
| User 4 | 10 | 9 | | 5 | |

<User-item rating matrix>

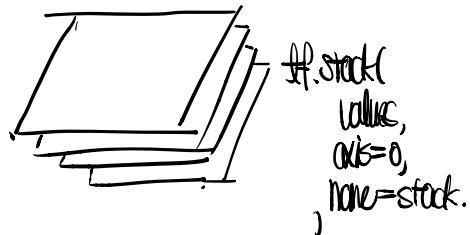
| Action | Sci-Fi | Comedy | Cartoon | Drama |
|---------|--------|--------|---------|-------|
| Movie 1 | 1 | 1 | | |
| Movie 2 | 1 | 1 | | |
| Movie 3 | | | 1 | 1 |
| Movie 4 | 1 | | 1 | 1 |
| Movie 5 | | | | 1 |

<item-feature matrix>

- for first user. "weighted feature matrix."

$$\begin{pmatrix} 4 & 4 & 0 & 0 & 4 \\ 6 & 6 & 0 & 0 & 9 \\ 0 & 0 & 88 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

... stack for every user.



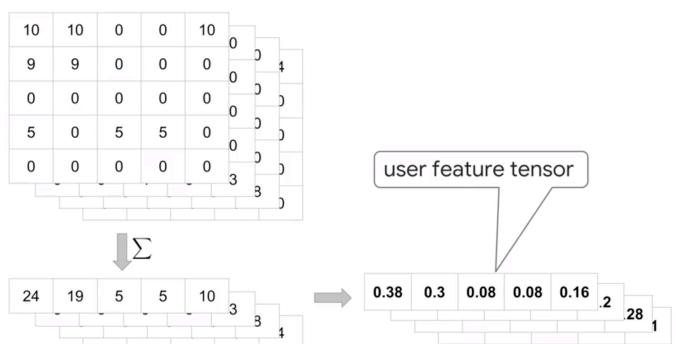
Quiz

What is the shape of the tensor resulting from stacking together all of the weighted feature matrices?

`tf.stack(wgtd_feature_matrices, axis = 0)`

- a) (# movies, # features)
- b) (# users, # movies, # features)
- c) (# features, #movies, #users)
- d) (# movies, # features, # users)

| | | | | | | | | | |
|----|----|---|---|----|---|---|---|---|---|
| 10 | 10 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 4 |
| 9 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 5 | 5 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



Which TensorFlow operation could we use to mask the previously rated movies in our user-movie ranking matrix, so we only focus on previously unrated movies when providing recommendations?

- a) tf.mask
- b) tf.strided_slice
- c) tf.tile
- d) tf.sparse_slice
- e) tf.where

Content based recommender aren't good at expanding the interest of users to new domains.