Joke Recommender System

Learning Portfolio 7

universität innsbruck

Collaborative Filtering

Collaborative filtering is a technique used in Recommender Systems, so that **past similar preferences of users inform future preferences.** It works by computing latent factors as the preferences of each user and collecting them in a vector called an **embedding.**

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	0	3	0	3	0
User 2	4	0	0	2	0
User 3	0	0	3	0	0
User 4	3	0	4	0	3
User 5	4	3	0	4	0

universität innsbruck

Training a collaborative filtering model

From the material provided, we have learned what a collaborative filtering model is, **how it can be trained** and how PCA can be used to interpret the user and movie embeddings.

Giving actual **recommendations** has not been shown to us practically.

```
ordinary_learn = collab_learner(dls, y_range=(0.0, 5.5))
ordinary_learn.fit_one_cycle(10, 5e-4, wd=0.1)
```

universität innsbruck

The Jester Dataset - 100 jokes and 25.000 users

The basis for the learning portfolio is this dataset here. It comprises the ratings of 25.000 users for 100 jokes. The texts for the jokes can be found here.

```
print(joke_ratings.head(5))
```

```
   user_id       1        2        3        4       5       6       7       8
0        0   0.545   4.6975   0.0850   0.4600  0.6200  0.3750  0.0375  3.5425
1        1   3.520   2.4275   4.0900   3.5925  1.9050  0.0850  2.3175  1.1650
2        2  27.250  27.2500  27.2500  27.2500  4.7575  4.8175  4.7575  4.8175
3        3  27.250   4.5875  27.2500  27.2500  2.9500  4.5400  1.7950  4.0525
4        4   4.625   3.6525   1.4575   1.1525  2.8400  2.9000  4.2600  3.6525
```

Computing cosine similarity

$$\frac{\sum_{i=1}^{n} cosinesimilarity_i rating_{ij}}{\sum_{i=1}^{n} cosinesimilarity_i}$$

We can **retrieve the embeddings** by calling the model attribute of our fastAI learner.

```
[17] loaded.model

     EmbeddingDotBias(
       (u_weight): Embedding(24984, 50)
       (i_weight): Embedding(101, 50)
       (u_bias): Embedding(24984, 1)
       (i_bias): Embedding(101, 1)
     )
```

Then, we can **compute the cosine similarity** between any user and the other users.

```
[47] user_factors = loaded.model.u_weight.weight
     similarities = nn.CosineSimilarity(dim=1)(user_factors[666], user_factors)[:-1]
     similarities.shape

     torch.Size([24983])
```

Computing a weighted average

$$\frac{\sum_{i=1}^{n} cosinesimilarity_i rating_{ij}}{\sum_{i=1}^{n} cosinesimilarity_i}$$

1. **Multiply** each user's i **rating** r with each **cosine similarity** between user x and user i
2. **Sum** all the **products** from 1 together
3. **Divide** by the **sum of cosine similarities**

We now predicted the rating r of user x for joke j

Computing a weighted average

$$\frac{\sum_{i=1}^{n} cosinesimilarity_i rating_{ij}}{\sum_{i=1}^{n} cosinesimilarity_i}$$

```
joke_ratings_without_user = joke_ratings.drop(["user_id"], axis=1)
joke_rating_for_joke_42 = torch.from_numpy(joke_ratings_without_user["1"].values)
joke_rating_for_joke_42 = torch.where(joke_rating_for_joke_42 > 5.1, torch.tensor(2.5), joke_rating_for_joke_42)
product = similarities * joke_rating_for_joke_42
product
```

```
sumproduct = torch.sum(product, dim=0)
sumproduct
```

```
[56] predicted_rating = sumproduct/similarities.sum()
     predicted_rating

     tensor(2.6388, dtype=torch.float64, grad_fn=<DivBackward0>)
```

universität innsbruck

Recommending jokes

$$\frac{\sum_{i=1}^{n} cosinesimilarity_i rating_{ij}}{\sum_{i=1}^{n} cosinesimilarity_i}$$

Now, instead of doing it for one item, we **predict** ratings **for all items** and select the highest predictions.

```
joke_ratings_without_user = torch.from_numpy(joke_ratings_without_user.values)
replaced_tensor = torch.where(joke_ratings_without_user > 5.1, torch.tensor(2.5), joke_ratings_without_user)
ratings = replaced_tensor * similarities[:, np.newaxis]
summated_ratings = torch.sum(ratings, dim=0)
weighted_ratings = summated_ratings/similarities.sum()
weighted_ratings.shape
```

```
values, joke_indices = torch.topk(weighted_ratings, 10)
joke_indices
```

```
tensor([49, 35, 31, 26, 34, 61, 28, 52, 48, 67])
```

Kontakt

Fabian Leuk
12215478
fabian.leuk@student.uibk.ac.at