Recommender Systems

Data Science Nigeria, 2018 Bootcamp

Robert John
@robert_thas

Recommender Systems

Brief History

- Man, and other creatures, have always ask for suggestions about something they want to do.
- A way of building on other peoples experiences.
- A way of deciding what I like based on item similarity.
- A way of deciding what I might like based on what someone else likes.
- Came to light because of Netflix Challenge.

Recommender Systems

Where do we see them

- Amazon: what to buy
- Netflix: what to watch
- Apple Radio: what to listen to
- Facebook: friends
- Medicine: repurpose drugs for diseases

Recommender Systems

Types

- Non-personalized recommenders
- Content-based recommenders
- Collaborative Filtering

Recommender Systems

Non-personalized Recommenders

- Popularity-based recommenders
  - What are people watching
  - What are people buying
- What are the advantages?
- What are the disadvantages?
- How do we implement these?

Recommender Systems

Non-personalized Recommenders

- What are the advantages?
  - Requires no ML!
  - Easy to implement with SQL
  - Scales massively
- What are the disadvantages?
  - Popular items become more popular
  - Less popular items never recommended
- How do we implement these?

sales.sql

```sql
SELECT
  SUM(hits_product_productQuantity) AS qty_sold,
  hits_product_v2ProductName
FROM
  `data-to-insights.ecommerce.rev_transactions`
GROUP BY
  2
ORDER BY
  1 DESC
LIMIT
  1000
```

Recommender Systems

User-based Recommenders

- Based on the user category
  - People in your category liked this
  - People in your category bought this
- Examples
  - People who bought this, also bought these
  - People who watched this, also watched that
- Advantages/Disadvantages

User ID	Property 1	Property 2	Property 3	Property 4
1	3	0	2	5
2	0	5	4	5
3	5	2	2	4
4	4	5	5	3
5	0	1	2	2
n	4	4	1	3

User ID	Item ID	Rating
1	1	4
1	2	3
2	2	5
2	47	5

user_based.py

```python
from sklearn.neighbors import NearestNeighbors


nn = NearestNeighbors(n_neighbors=5, radius=2.0)
nn.fit(users)


user = np.array([1, 4, 4, 5])
d, neighbors = nn.kneighbors(user.reshape(1, -1))


print(neighbors)
```

predict.py

```python
suggested_products = []

for n in neighbors:
 for products in user_products[n]:
   for product in products:
     if product != 0 and product not in suggested_products:
     suggested_products.append(product)

print(suggested_products)
```

Recommender Systems

Content-based Recommenders

- Based on the actual content
  - Explicit comparisons (action, comedy, romance)
  - Implicit comparisons (factor matrix)
  - Word comparisons
- Examples
  - Articles similar to what you are reading
  - Movies similar to what you are watching
- Advantages/Disadvantages

Recommender Systems

Collaborative Filtering

- Social filtering
  - Uses recommendations from other users
  - Uses ratings (explicit or implicit)
- Examples
  - People who rated this article the same as you also liked these articles
  - Peole who rated this song the same as you also liked these songs
- Advantages/Disadvantages

collaborative.py

```python
import pandas as pd
import numpy as np
from tensorflow import keras

ratings = pd.read_csv('./train.csv')
test_df = pd.read_csv('./test.csv')
test_df['Rating'] = 0

merged_temp = pd.concat([ratings, test_df], axis=0)
merged_temp.reindex()
```

collaborative.py

```
viewers = merged_temp.Viewers_ID.unique()
jokes = merged_temp.Joke_identifier.unique()

viewer_min, viewer_max, joke_min, joke_max = \ (merged_temp.Viewers_ID.min(),
merged_temp.Viewers_ID.max(),\ merged_temp.Joke_identifier.min(),
merged_temp.Joke_identifier.max())

n_viewers = merged_temp.Viewers_ID.nunique()
n_jokes = merged_temp.Joke_identifier.nunique()
N_FACTORS = 32
np.random.seed = 42
REG_STRENGTH = 1e-9
```

collaborative.py

```python
from keras.layers import Input, Embedding, Flatten, merge
from keras.regularizers import l2
from keras.optimizers import Adam
from keras import Model

def create_embedding(name, n_in, n_out, reg):
    inp = Input(shape=(1,), dtype='int64', name=name)
    emb = Embedding(n_in, n_out, input_length=1,
embeddings_regularizer=l2(reg))(inp)
    return inp, emb
```

collaborative.py

```python
def create_bias(inp, n_in):
    #Flatten()(Embedding(n_in, 1, input_length=1)(inp))
    e = Embedding(n_in, 1, input_length=1)
    x = e(inp)
    x = Flatten()(x)
    return x


from keras.preprocessing.text import Tokenizer
```

collaborative.py

```
view_tok = Tokenizer(num_words=n_viewers + 2, lower=False)
view_tok.fit_on_texts(viewers)
print('Found {} unique tokens for viewers.'.format(len(view_tok.word_index)))


joke_tok = Tokenizer(num_words=n_jokes + 2, lower=False, split='|')
joke_tok.fit_on_texts(jokes)
print('Found {} unique tokens for jokes.'.format(len(joke_tok.word_index)))
```

collaborative.py

```python
def get_viewer_idx(df, idx):
    df['viewers'] = df.Viewers_ID.apply(lambda x: idx[x])
    return df

def get_joke_idx(df, idx):
    df['jokes'] = df.Joke_identifier.apply(lambda x: idx[x])
    return df
```

collaborative.py

```
ratings = get_viewer_idx(ratings, view_tok.word_index)
ratings = get_joke_idx(ratings, joke_tok.word_index)

msk = np.random.rand(len(ratings)) < 0.8
trn = ratings[msk]
val = ratings[~msk]

viewer_in, v = create_embedding('viewer_in', n_viewers + 2, N_FACTORS,
REG_STRENGTH)
joke_in, j = create_embedding('joke_in', n_jokes + 2, N_FACTORS, REG_STRENGTH)
```

collaborative.py

```
x = merge([v,j], mode='dot')
x = Flatten()(x)
dot_model = Model([viewer_in, joke_in], x)
dot_model.compile(adam, loss='mse')

dot_model.fit([trn.viewers, trn.jokes], trn.Rating, batch_size=64, epochs=3,
validation_data=([val.viewers, val.jokes], val.Rating))
```

collaborative.py

```
predictions = dot_model.predict([test.viewers, test.jokes])



out_file = test[['Response_ID']]
out_file['Rating'] = predictions
out_file.head()
```

Going forward

Google Cloud Platform

cloud.google.com/

TensorFlow

tensorflow.org

scikit-learn

scikit-learn.org

Thank You!

Robert John
@robert_thas