Google Developers Experts

Guardian News Recommendation

Data Science Nigeria, 2018 Bootcamp

Robert John
@robert_thas

Collaborative Filtering

What others similar to me are reading

Reader ID	Article 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)
```

Recommending News

Content-based Recommender

News Recommender

Content-based Recommender

- Analyze contents of documents
- Pick a distance measure
- Pick a document
- Compute distance of other documents from that document
- Rank documents based on nearness
- Recommend top-n documents

News Recommender

Analyze contents of documents

- Find a numeric representation
  - Create a dictionary
- Find an encoding
  - Bag of Words
  - Term-Frequency Inverse Document Frequency
  - Embeddings

News Recommender

Pick a distance measure (Nearest Neighbors)

- Manhattan Distance: |a - b|
- Euclidean Distance: (a - b) ^ 2
- Pearson Correlation Coefficient
- Cosine Similarity

News Recommender

Compute Distances

- Create A Matrix
- Compute Distances

News Recommender

Rank & Recommend

- Pick an Item (Column)
- Get neighbors (Rows)
- Sort by order of nearness
- Show top n.

Recommending News

Collaborative Filtering

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

```python
from keras.layers import Dense

viewer_in, v = create_embedding('viewer_in', n_viewers + 2, N_FACTORS, 1e-5)
joke_in, j = create_embedding('joke_in', n_jokes + 2, N_FACTORS, 1e-5)

x = merge([v,j], mode='concat')
x = Flatten()(x)
x = Dense(128, activation='relu')(x) # overfit on dropout and add
regularization
x = Dropout(0.5)(x)
x = Dense(256, activation='relu')(x)
x = Dense(1)(x)
```

collaborative.py

```
wide_model = Model([viewer_in, joke_in], x)
wide_model.compile(adam, loss='mse')

wide_model.fit([trn.viewers, trn.jokes], trn.Rating, batch_size=256, epochs=3,
validation_data=([val.viewers, val.jokes], val.Rating))

predictions = wide_model.predict([test.viewers, test.jokes])
out_file = test[['Response_ID']]
out_file['Rating'] = predictions
out_file.head()
```

collaborative.py

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



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

Challenges

Handling Real-world Data at Scale

Service	Subscribers	Catalog Size
Apple Music	40M	40+M
Spotify	80M	30+M
Netflix	130M	6.5+K
Youtube	1.8B	5B

Scaling Problems

- New subscribers joining
- Updating catalog
- Keeping the model up-to-date

Approach	Viability
Singular Value Decomposition	Fails
Matrix Factorization	Struggles
Latent Factorization	Works
Neural Networks	Works
kNN	Works
Alternating Least Squares	Favored

Framework	Viability
Scikit-Learn	Fails
PyTorch	Works
TensorFlow	Works
PySpark	Works

Solutions

Handling Real-world Data at Scale

Scaling Problems

- Store massive amounts of training data (Google Cloud Storage)
- Read out-of-memory data (tf.data)
- Process data in parallel (Spark/Beam on Cloud Dataflow)
- Distributed Training (TensorFlow Estimators and Google Cloud Machine Learning Engine)
- Faster Vectorization (Cloud Tensor Processing Units)
- Hyper-Parameter Tuning (Cloud ML Engine)
- Serving at scale (Cloud ML Engine)

Going forward

TensorFlow

tensorflow.org

Google Cloud Platform

cloud.google.com/

Google Cloud Machine Learning Engine

https://cloud.google.com/ml-engine/

Google Cloud Dataflow

https://cloud.google.com/dataflow/

Thank You!

Robert John
@robert_thas

Google Developers Experts