kNNでおすすめの映画を取得するモデルを実装【②ユーザベースレコメンド】

2020.03.24 2020.02.24

Python

以前の記事で、映画のMovielensのデータとkNNを使って、同一ユーザに別の映画をレコメンドする仕組みを実装しました。

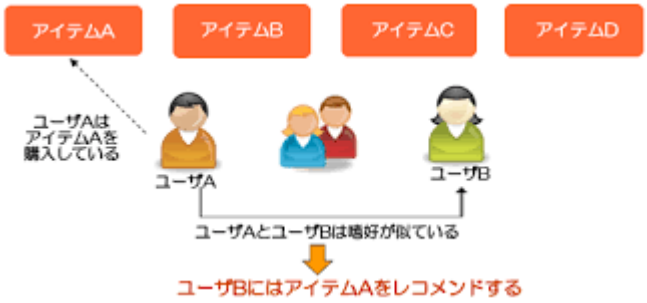
kNNでおすすめの映画を取得するモデルを実装【①アイテムベースレコメンド】省メモリでスパース行列を扱うテクニックも説明します

今回はレコメンドエンジンの代表的なアルゴリズムであるkNNを実装していきます。kNN（kNearestNeighber、k近傍）とはkNNとは、空間（平面）上にマップされたサンプル郡から相対距離を求めるものです。分...

career-tech.biz 2020.02.22

今回は前回と同じ形式のデータを使い、逆に『ユーザの履歴から、別のユーザに同じアイテムをレコメンド』します。イメージは下記にお借りした図のとおりです。

『この映画を見ている人はこんな映画も見ています』という表現は同じですが、背後のアルゴリズムというかデータの構造が異なるわけです。



https://www.activecore.jp/column/rwh_2/

Pythonでの実装開始

実装していきましょう。前回と同じくMovielensのデータですが、前回は200万件のデータを使ったのに対し、今回は予め100k（10万件）のデータを使います。下手に加工するより偏りがないためです。

> **MovieLens**
>
> GroupLens Research has collected and made available rating data sets from the MovieLens web site (The data sets were collected over various periods of time, de...
>
> grouplens.org

前処理

では、下記にコードです。

```
import pandas as pd
import numpy as np
from sklearn.neighbors import NearestNeighbors
from scipy.sparse import csr_matrix
import time
import matplotlib.pyplot as plt

#データを読み込む
print('Loading dataset...')
train = pd.read_csv("/Users/yumatakenaka/Data/ub.base", names=["userId", "movieId", "rating", "t
test = pd.read_csv("/Users/yumatakenaka/Data/ub.test", names=["userId", "movieId", "rating", "ti
print('Finished')

# カラムをカテゴリ変数化
```

```
userId_categorical = pd.api.types.CategoricalDtype(categories=sorted(train.userId.unique()), ordered
movieId_categorical = pd.api.types.CategoricalDtype(categories=sorted(train.movieId.unique()), ord
# カテゴリインスタンスを利用して新しいカラムを作成
row = train.userId.astype(userId_categorical).cat.codes
col = train.movieId.astype(movieId_categorical).cat.codes
# マトリックスにRatingの数値を当てはめる
sparse_matrix = csr_matrix((train["rating"], (col, row)), shape=(movieId_categorical.categories.size
# スパース行列をDataframe化する
train_pivot = pd.SparseDataFrame(sparse_matrix, index = movieId_categorical.categories, columns =
```

一気に前処理まで終えました。前回と異なるところは一点のみで、最終的に作るデータのrowとcolを入れ替えた点です。

元データは完全に縦持ちのデータになっているため、いずれにせよ行と列を別々に取り出す作業は必要になります。

```
# n_neibors やalgorithm、metricなど重要なアーギュメントを設定
knn = NearestNeighbors(n_neighbors=9,algorithm= 'brute', metric= 'cosine')
# 前処理したデータセットでモデルを訓練
model_knn = knn.fit(train_pivot)
```

近いユーザの好む作品を提案

そしてFitさせます。ここからが少しだけ異なります。

というのも、ユーザに対して「あなたと似ているユーザーはこんな人です」と表示しても意味がないため、近しいユーザを推論した上で更にそのユーザが好む映画を提案していきます。

```
def limited_list(elements, n):
    list = []
    for num in range(n):
        list.append(elements[0])
        del elements[0]
    return list

def movie_prediction(movie):

    distance, indice = model_knn.kneighbors(train_pivot.iloc[train_pivot.index== movie].values.reshap
    for i in range(0, len(distance.flatten())):
        if  i == 0:
```

```
        print('to ID {0}, Here are the similar users to you\n'.format(train_pivot[train_pivot.index == m
    else:
        print('{0}: {1} with distance: {2}'.format(i,train_pivot.index[indice.flatten() [i]],distance.flatte
        result = train_pivot[train_pivot.index[indice.flatten() [i]]]
        result_sorted = sorted(result.items(), reverse=True, key=lambda x:x[1])
        result_final = limited_list(result_sorted,3)
        print(result_final)
```

関数を2つ作っていて、下のほうは推論をして、該当するユーザの好みの映画をレーティング点数で降順に並び替え、戻り値として渡します。

下の関数で呼ばれる上の関数ではソートされたリストから上位n個の要素のみを返す関数です。意外と組み込みに無かったため簡単ですが作りました。

さて、下の関数にユーザIDを渡して推論を返してみましょう。今回はID==127としています。

```
to ID 127, Here is similar users to you
1: 50 with distance: 0.34652532086052235
[(253, 5), (475, 5), (1084, 5)]
2: 187 with distance: 0.37896009942266407
[(8, 5), (64, 5), (65, 5)]
3: 100 with distance: 0.4002401471950543
[(313, 5), (315, 5), (258, 4)]
```

こちらが結果です。

ユーザ127に対し、最もユークリッド距離が近いのがユーザ50で、更にユーザ50がオススメするベスト3が映画ID253,475,1084で、全てに最高点数の5点をつけています。

同様に、3番目までに近いユーザの、それぞれ3つおすすめの映画を取り出しています。

まとめ

今回は、前回に引き続きMovielensのデータを使って、ユーザ同士の距離と、ペルソナの似たユーザのおすすめ映画を提案しています。

私自身やっていて疑問に思ったのですが、一つのデータ、アルゴリズムで2つの解釈をすることができます。
では、アイテムベースとユーザベースの強調フィルタリングではどちらが有用なのでしょうか。

サイト内レコメンドにおける推薦アルゴリズム | PINTO! by PLAN-B

近年、色々なサイトで見られるようになった、”レコメンド”。「この商品を買った人は、こちらも購入しています。」「あなたにおすすめの記事」など、ECサイトだけでなくブログにまで存在し、サイトに訪問した …

service.plan-b.co.jp

上記のブログでは「一般的に世の中のインターネットサービスは膨大なユーザーを抱えており、各ユーザーの趣味・嗜好は動的に変化」することから、変化の大きなユーザベースよりもアイテムベースのほうがワークする可能性が高いことが述べられています。

一方で、（私もそうですが）相対的に少品種のBtoB企業においては、単にレコメンド以外にも自分たちの顧客同士の相対距離を知ることで新たなインサイトになる可能性もあるなと思いました。

特に営業的な戦略としてはユースケースや課題感が共通していれば、他社事例などがかなり強力に使えるのではないでしょうか。

2つの方法で感じ方も異なってくるので、今後もじっくりと検証していきたいです。

今回は以上です。お読みいただきありがとうございました。