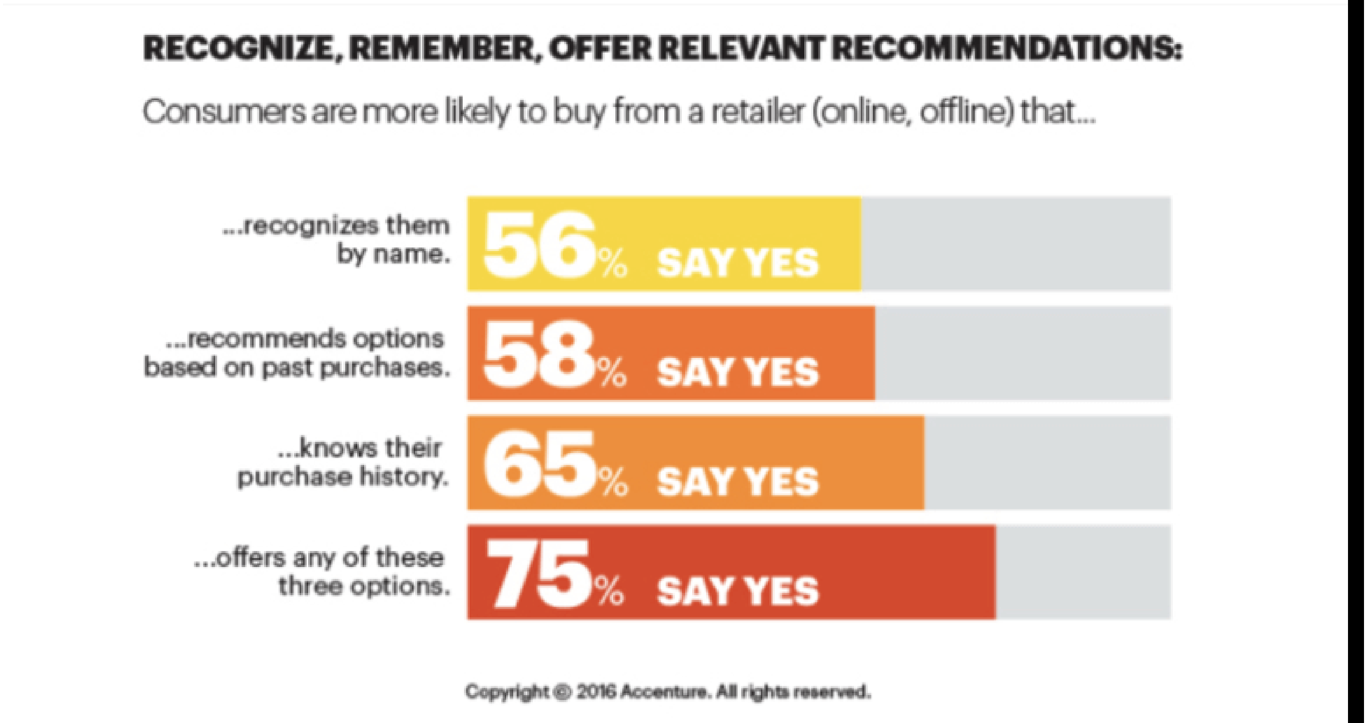
Alternating Least Square for Implicit Dataset with code

A guide to recommendation engines for feedback(interaction) data

Himanshu Kriplani Follow
Jun 26, 2019 · 5 min read

In the era of big data and analytics, the power which a data driven business has been exponentially increasing. Greater integration of AI and Machine learning has played a vital role in development of systems which can benefit both the Business and Business user as well. Moreover,Recommendation systems add an edge to digital businesses. Below chart shows the importance of recommendation systems.



Importance of Recommendation engines

A dataset containing explicit rank, count or category of particular item or event is considered an explicit data item. 4 out of 5 rating of a movie is an explicit data point. Whereas, in implicit dataset we need to understand the interaction of users and/or events to find out its rank/category. For example, a person watching a single genres of the movies. This kind of datasets are considered implicit. We had miss a whole lot of hidden insights if we don't consider implicit datasets.

而在隐式数据集中，我们需要了解用户和/或事件的交互作用以找出其等级/类别。 例如，一个人观看一部电影。 这种数据集被认为是隐式的。 如果不考虑隐式数据集，我们将错过很多隐藏的见解

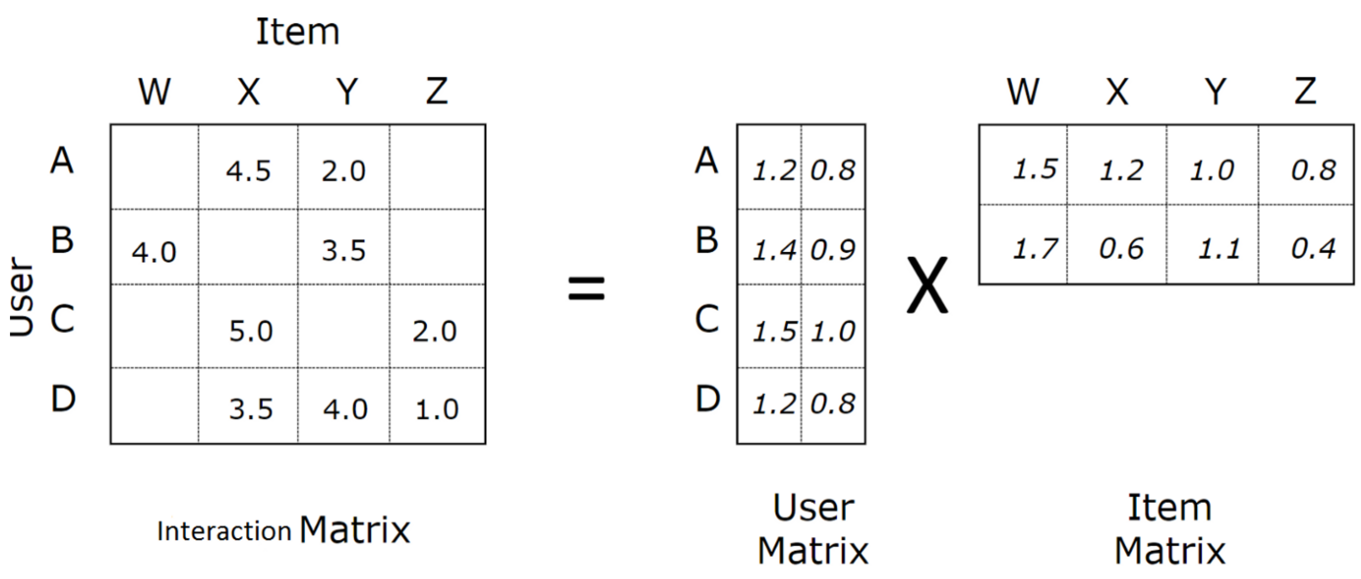
隐式数据集仅包含用户和项目交互

> ***Implicit dataset contains user and item interactions only***

Further we will take in consideration an example to have a better glimpse. Retailrocket recommender system dataset is one very good implicit data to understand Alternating least square method.

Alternating Least Square

Alternating least square method is an algorithm to factorize a matrix.We will discuss how Collaborative Filtering for Implicit Feedback Datasets uses ALS. As in the below figure, we see that a matrix being factorized into 2 smaller matrices. Consider the first matrix as the set of user-item interaction. Thus, the factorized matrices would be user features and item features



交互矩阵的值是事件　　　　特定偏好和信心

The Interaction matrix values are events which has specific preferences and confidence, which give the value of the each element. For an instance consider E-commerce dataset

查看，添加到购物车并进行交易

which has 3 events: View, Add-to-cart and Transacted. If there is an interaction between the User and Item pair, it is considered positive preference else negative.

交互矩阵值是具有特定首选项和置信度的事件，这些事件给出每个元素的值。 例如，考虑具有3个事件的电子商务数据集：查看，添加到购物车和交易。 如果“ 用户 ”和“ 项目 ”对之间存在交互，则认为是积极的偏好，否则是消极的

$$P_{ui} = \begin{cases} 1 & R > 0 \\ 0 & R = 0 \end{cases} \text{ where } R = r_{ui}$$

Preference

> *Confidence can be defined as the worth or the value we give to the interaction. For User A buying(a transaction event) item X we increase the interaction weight, while User A viewing item Z has lesser weight than the 'interaction of buying'.*

信心可以定义为我们给予互动的价值或价值。对于用户A购买(交易事件)物品X，我们增加了交互权重，而用户A观看物品Z的交互权重小于购买的交互权重。

$$C_{ui} = 1 + \alpha r_{ui}$$

Confidence

Confidence: r is the interaction between User u and Item i. More the interaction, more the confidence — scaled on the value **α.** A item bought 5 will have more confidence than the item bought just twice. We add 1 incase r is 0 for that interaction making it nonzero. The paper typically recommends 40 as the **α** value.

本文介绍了以下成本函数，用于查找用户交互和项目交互矩阵

The paper describes about the following cost function for finding User interaction and Item interactions matrices:

$$\min_{y*,y*} \sum_{u,i} C_{ui} P_{ui} - X_u^T Y_i)^2 + \lambda \left(\sum_u || x_u ||^2 + \sum_{u,i} || y_i ||^2\right)$$

Here, λ is used for regularizing the model and its value can be determined by cross validation.

交替最小二乘的本质

The essence of Alternating Least Square

The cost function contains m · n terms,where m is the number of users and n is the number of items. For typical datasets m · n can easily reach a few billion. Thus,

因此，随机梯度下降等优化方法会对如此庞大的数据造成混乱。因此，本文介绍了可选优化技术

opitimization methods such as Stochastic gradient descent would make a mess for such huge data. Therefore, paper introduces alternative optimization technique.

> *Observe that when either the user-factors or the item-factors are fixed, the cost function becomes quadratic so its global minimum can be readily computed. This leads to an alternating-least-squares optimization process, where we alternate between re-computing user-factors and item-factors, and each step is guaranteed to lower the value of the cost function*

We find the user(x) vector and the item(y) vector by differentiating the above cost function by x and y respectively.

$$X_u=(Y^TC^uY+\lambda I)^{-1}Y^TC^up(u)$$

$$Y_i=(X^TC^iX+\lambda I)^{-1}X^TC^ip(i)$$

User and Item Vectors

So, now to find a preference score for user-item pair we use the following:

$$p_{ui} = x^Ty_i$$

We find the largest p value having items to recommend to an user.

How do we code it?

We use Implicit Library which has all the implementations in Cython making it faster.

> ***Import Libraries***

```
import pandas as pd
import numpy as np
import scipy.sparse as sparse
import random
import implicit
```

Data Preprocessing

Dont worry[This part need not be same for every use case]

```
def create_data(datapath,start_date,end_date):
    df=pd.read_csv(datapath)
    df=df.assign(date=pd.Series(datetime.fromtimestamp(a/1000).date()
for a in df.timestamp))
    df=df.sort_values(by='date').reset_index(drop=True) # for some
reasons RetailRocket did NOT sort data by date
    df=df[(df.date>=datetime.strptime(start_date,'%Y-%m-%d').date())&
(df.date<=datetime.strptime(end_date,'%Y-%m-%d').date())]
    df=df[['visitorid','itemid','event']]
    return df

datapath= './input/events.csv'
data=create_data(datapath,'2015-5-3','2015-5-18')
data['user'] = data['user'].astype("category")data['artist'] =
data['artist'].astype("category")data['user_id'] =
data['user'].cat.codesdata['artist_id'] = data['artist'].cat.codes
```

Creating Interaction Matrices

As the data is sparse, we create sparse matrix for item-user data which will be an input to the model. user-item matrix will be used for getting recommendations

```
sparse_item_user = sparse.csr_matrix((data['event'].astype(float),
(data['itemid'], data['visitorid'])))

sparse_user_item = sparse.csr_matrix((data['event'].astype(float),
(data['itemid'], data['visitorid'])))
```

The ALS

```
#Building the model
model = implicit.als.AlternatingLeastSquares(factors=20,
regularization=0.1, iterations=20)

alpha_val = 40
data_conf = (sparse_item_user * alpha_val).astype('double')

model.fit(data_conf)
```

Using the Model

Getting the **recommendations** using the inbuilt library function

```
#Get Recommendations

user_id =   14

recommended = model.recommend(user_id, sparse_user_item)

print(recommended)
```

We can also have a list of **similar items** using the following function

```
#Get similar items

item_id = 7

n_similar = 3

similar = model.similar_items(item_id, n_similar)

print(similar)
```

Conclusion

Alternating least square method is pretty robust. The time complexity for ALS is in the linear scale to the size of the data. (i.e. **O(x*n) where n is the length of the dataset**

and x is an integer). The code is available at this link