Implicit feedback Recommender System

Thibault ALLART

Collaborative filtering

https://buildingrecommenders.wordpress.com/2015/11/18/overview-of-recommender-algorithms-part-2/

Issue 1: item set is not static (new products every days)

What is the prediction for a new product ?

Solution 1: content based recommendation

	Pure Colaborative filtering	Content based
 Tom	u5342761	Male, 35, living in Barcelona, has bought 1 book on deep learning last month, ...
	i87239	A book, python, deep learning, 40 €, released in 2017, ...

Issue 2: hard to add new features

4	3			5	
5		4		4	
4		5	3	4	
	3				5
	4				4
		2	4		5

What if we want to add more information into this model?

You can add categorical data but the matrix increase exponentially.

Even worse for continuous data.

Examples:

- How to include brand similarity?
- How to include temporality?
- How to add contextual information?

Solution 2 : Easier with the regression (DL) point of view

img: https://github.com/m2dsupsdlclass/lectures-labs

Issue 3: Time matters

- Real data is not a rating matrix but an ordered list of events.

user_id	item_id	rating	timestamp
196	242	3	881250949
186	302	3	891717742
22	377	1	878887116
244	51	2	880606923
166	346	1	886397596

Time is lost

4	3			5	
5		4		4	
4		5	3	4	
	3				5
	4				4
		2	4		5

Solution 3: Time matters

Bad way

Good way

Issue 4: we don't have scores

4	3			5	
5		4		4	
4		5	3	4	
	3				5
	4				4
		2	4		5

User doesn't give us a score for each product.
We don't have explicit feedback but only implicit ones.

What value should we put into the matrices ?

50*purchase + 12*add_to_cart + 3.1*View_product_page

Why not 3.2 or Pi ?

Solution 4: Impose **rank** not value

Instead of forcing a model to fit a prior scalar value,
let the model free to assign any scalar value it want,
as far as it respect an order relationship.

For a given user, we want the score S with an item he will buy (item+) be higher than the score for an item he will not buy (item-).

$$S(\text{user}, \text{item}+) > S(\text{user}, \text{item}-)$$

Model

Model : siamese network with triplet loss

Training

1

```
generate_positive_sample(purchase_table):
    (user, item+, timestamp) = select a random purchase
    return (user, item+, timestamp)
```

2

```
generate_negative_sample(timestamp, item+, item_table):
    candidates = items at sales at timestamp remove item+ from candidates
    item- = select a random candiate return item-
```

Training

3

```
features(user, item, timestamp, tables)
 # features can be any function that depend on user, item and/or timestamp
 # providing its doesn't include information at time >= timestamp. i.e
    user_age
    item_brand
    item_ranking_in_the_last_hour
    number_of_time_user_bought_this_item_category

    return features
```

Prediction

1. For a given (user, timestamp) build a batch with all open operations (items).
2. Predict the score for the whole batch.

3. Return operations sorted by decreasing score.