A simple and effective method for RecSys Challenge 2022

Juntong Ni
Shandong University
Qingdao, China

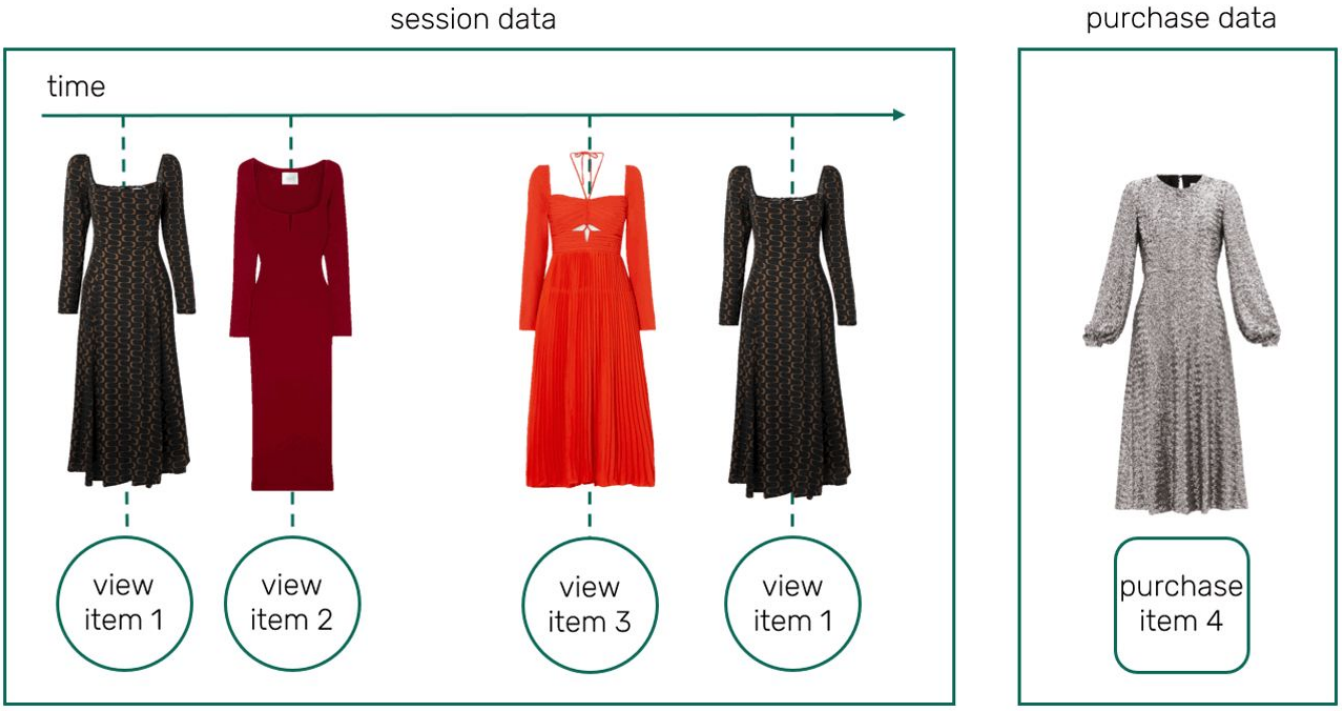

Figure 1: Example Session and Purchase Data.

ABSTRACT

Although deep learning models can do very well on many tasks, including recommendation systems, the problem of computationally intensive and poorly interpretable models is still difficult to solve. In this paper, we apply the reduced storage complexity data preprocessing method and KNN algorithm to the field of recommendation systems to solve the challenges of RecSys2022, which takes less than 50 minutes to run on CPU and achieves a final score of about 0.169 on leaderboard, which can show that the method is simple and effective.

Final Project, Machine Learning and Pattern Recognition, Shandong University

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

CCS CONCEPTS

• **Information systems** → *Task models*; *Users and interactive retrieval*.

KEYWORDS

RecSys datasets, machine learning, sparse matrix, KNN

ACM Reference Format:
Juntong Ni. 2022. A simple and effective method for RecSys Challenge 2022. In *Proceedings of CS (Final Project).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 CHALLENGE TASK

This year's challenge focuses on fashion recommendations. When given user sessions, purchase data and content data about items, can you accurately predict which fashion item will be bought at the end of the session? The content data consists of descriptive labels of the items (such as color, length, neckline, sleeve style, etc.). The labels have been assigned using Dressipi's human-in-the-loop system where fashion experts review, correct and confirm the correctness of the labels, so we expect this to be a dataset of high accuracy and quality.

2 DATASET VISUALIZATION

The full dataset consists of 1.1 million online retail sessions in the fashion domain, sampled from a 18-month period. All sessions in the dataset are "purchasing sessions" that resulted in at least one item purchased. The items viewed and purchased are clothing and footwear. The dataset contains content data for each of the items viewed and purchased, this is an extract of Dressipi's fashion item data and represents descriptive labels assigned to the items such as color, neckline, sleeve length etc. The task is to predict the item that was purchased. You could get access to dataset with http://www.recsyschallenge.com/2022/index.html#participation.

* Sessions: The items that were viewed in a session. In this dataset a session is equal to a day, so a session is one user's activity on one day.
* Purchases: The purchase that happened at the end of the session. One purchased item per session.
* Item features: The label data of items. Things like "color: green," "neckline: v-neck," etc.

train_purchases.csv contains purchase records of different sessions in 14 months, with a total of 1,000,000 records. The purchase that happened at the end of the session. One purchase per session. After code check, no NaN or NULL data is found, and no duplicate data is found.

train_sessions.csv stores 4743820 past browsing records. The items that were viewed in a session. The "date" column is a timestamp to miliseconds. A session is equal to a day, so a session is one user's activity on one day. The session goes up to and not including the first time the user viewed the item that they bought in the end. The last item in the session will be the last item viewed before viewing the item that they bought. To find they item they bought link to train_purchases.csv on session_id. After code check, no NaN or NULL data is found, and no duplicate data is found. Average items seen during session is 4.74382. Figure 2 shows the distribution of the number of 100 commodities, and Figure 3 shows the distribution of the number of 100 commodities as a percentage of the total. It can be seen that there is a large variation in the distribution of the number of commodities, and there is an obvious long tail problem. This is also a common problem in recommendation systems, where some items are purchased by more people, but some items are rarely purchased, and it is challenging to train the model to solve this problem. Figure 4 is a box plot, you can see that a small number of items account for a large percentage of the total, and there are many items with very small quantities

item_feature.csv contains the feature information of all items, with a total of 471,751 records. The label data of items. A feature_category_id represents an aspect of the item such as "colour", the feature_value_id is the value for that aspect, e.g. "blue". Some items may not share many feature_cateogry_ids if they different types of items, for example trousers will share almost nothing with shirts. Even things like colour will not be shared, the colour aspect for trousers and shirts are two different feature_category_ids. The smallest feature id is 1, the largest feature id is 73, each item has 19.9 features on average, each item has at least 2 features and at most 33 features. Different items have different numbers of features. The horizontal coordinate of Figure 5 is the number of features and the vertical coordinate is the number of products, which means that the number of products with 24 features is the largest, and the overall distribution is approximately normal. There are 73 features in the dataset, and Figure 6 shows for each feature, what is the number of goods that have this feature. It can be seen from this that the number of goods corresponding to different features varies greatly.

test_leaderboard_sessions.csv stores the test data, which has 229354 records. The distribution of each item is also counted, and Figure 7 shows the proportional distribution of the number of the top 20 items per session only.

Figure 2: Number of times each item appears.

Figure 3: Distribution of the number of times each item appears in the total

3 METHODOLOGY

The method is divided into two parts, one introducing our efficient data preprocessing method and the other describing how we use the nearest neighbor algorithm. There are no complicated formulas here, as these algorithms are very basic.

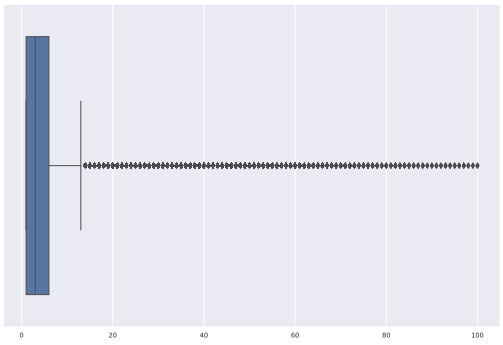

Figure 4: Box plot of item quantity distribution.

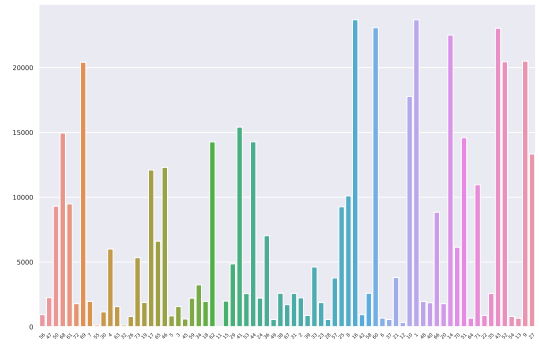

Figure 6: The distribution of the number of items corresponding to different features.

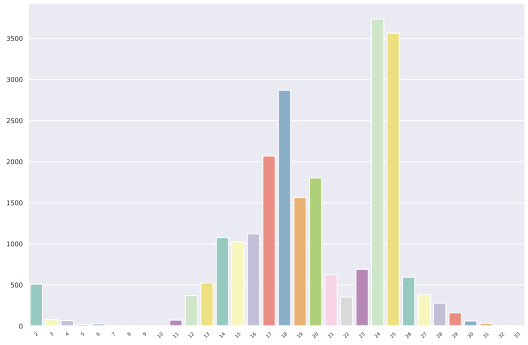

Figure 5: Distribution of the number of items with different number of features.

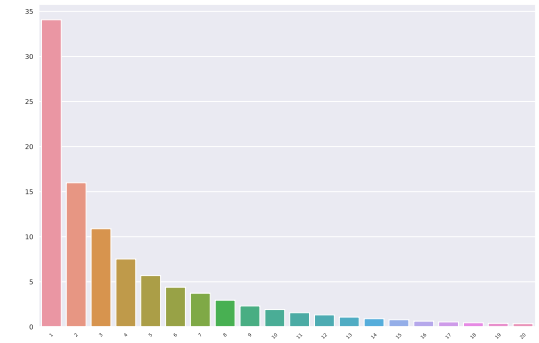

Figure 7: The proportional distribution of the number of items per session in the leaderboard.

3.1 Data Preprocessing

One of the frequent data preprocessings when doing machine learning and statistical analysis is the vertical and horizontal conversion of data.

Vertical and horizontal conversion refers to converting vertical (or long) data and horizontal (or wide) data to each other.

When predicting user behavior, we create features from the user's behavior logs such as the browsing history of the item, but in RecSys2022 dataset, such behavior logs are kept in a vertical format in tables on Hadoop and parallel RDB.

However, when using that data for machine learning processing, it will be converted into a horizontal format where one record becomes one sample, and then put it into model learning and predictive processing.

We will consider how it can be used to target large-scale data, especially when converting vertical data to landscape.

When writing a program in Python, you often manipulate data using the data frame format of pandas, but the pandas also has functions for vertical and horizontal conversion.

If you convert from vertical to horizontal, the number of columns in the table tends to increase, so there is a possibility that the data after conversion (horizontal holding format) will consume a large amount of memory. Also, depending on the machine specifications, there may be not enough memory and the conversion itself may not be possible. Therefore, you need to be a little careful, especially when converting large-scale data to landscape.

If the data converted to horizontally is a so-called sparse matrix (a matrix where most cells are 0), the output of the horizontally-bearing process is pandas.SparseDataFrame instead of pandas.DataFrame. By doing so, you can save a lot of memory. We finally use pandas.Categorical to deal with it. The details of the category data of pandas are described in https://pandas.pydata.org/pandas-docs/stable/user_guide/categorical.html

So, as a way to convert vertical data to memory saving, we use pandas.Categorical. This method does not go through ordinary horizontal data frames or dense matrixes in the middle of processing, so we were able to get the result of horizontally converting without spending a lot of memory.

Table 1

Hyperparameter	Value	Comments
n_neighbors	10	Number of neighbors
radius	1	Range of parameter space to use
algorithm	brute	Compute the nearest neighbors
metric	cosine	The distance metric

3.2 Nearest Neighbors Algorithm

When shopping on the global EC site "Taobao.com", we often see the indication that "the person who bought this product also buys this product." I often buy the products proposed in this way on Taobao.com. Certainly, I feel like you are proposing something close to the product I want.

A technology called collaborative filtering is used to realize such a mechanism. Cooperative filtering suggests recommended products based on similarity. Recommending (recommendation) products based on similarity of user preferences (purchase trends) is called user-based, and recommendation based on product similarity is called item-based. It's easy to get an image of the similarity of the product, but it's difficult to understand the image of the similarity of the user's preferences.

With user-based recommendations, while we can propose products tailored to individual users' preferences, we can't respond to new users without purchasing history or new products that have not been purchased by anyone.

Conversely, item-based recommendations (recommendations) allow you to propose products to new users who do not have a purchase history, while the same product is proposed to all users.

By the way, we use cosine similarity to calculate similarity. Cosine similarity is a mechanism to calculate similarity with something like an "angle". However, as with KNN, which calculates similarity based on distance, it is affected by the curse of the dimension. For KNN and the curse of the dimension, please refer to the following article.

It's easy, but I'll explain the curse of the dimension in cooperative filtering. Think of dimension as the number of characteristics. Characteristics are attributes of data. Human attributes include age, height, and weight. As for the attributes of the product, the category, size, price, etc. are the same. When judging whether it is similar by cosine similarity, if the number of features is small, the similarity tends to be higher, and if the number of characteristics is large, the similarity tends to be low. Therefore, it is necessary to devise ways to avoid the curse of the dimension.

Therefore, there are recommendation algorithms such as Matrix Factorization, which avoids the curse of the dimension.

From here, we will create an AI that recommends (recommends) recommended items. Specifically, we implement item-based collaborative filtering using the programming language Python. We use KNN (K-neighbhood method)[2][1] as an algorithm. You could get more details about KNN in https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html

Id	Notes	Score	Uploaded at	Rank
e57013b1-7609-4005-aaa3-5c7c8f64d195		0.16856950421774286	29 May 07:41	1604

Figure 8: My result in the leaderboard.

4 RESULT

We conducted experiments using the hyperparameters in Table 1 and finally obtained a score of **0.16856950421774286** on the leaderboard test dataset. To prove the veracity of my results, I show it in Figure 8.

5 CONCLUSION

For the task of recommendation system as well as the dataset, we propose a memory-saving data preprocessing method based on matrix sparsification and a recommendation system based on KNN algorithm, our method is a traditional machine learning method, compared to deep learning methods, we have a clear advantage in computational complexity, and at the same time the method has achieved very beneficial results. In the future, more efficient algorithms may be explored to solve the problem.

ACKNOWLEDGMENTS

Thank you to Dr. Xin Xin for teaching me Machine Learning and Pattern Recognition for one semester, which broadened my horizon and I learned a lot, which helped me to finish this big assignment successfully. I would also like to thank Shuyu Guo, Jiyuan Yang and Yulong Huang for their hard work and dedication.

REFERENCES

[1] Naomi S Altman. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician* 46, 3 (1992), 175–185.
[2] Evelyn Fix and Joseph Lawson Hodges. 1989. Discriminatory analysis. Nonparametric discrimination: Consistency properties. *International Statistical Review/Revue Internationale de Statistique* 57, 3 (1989), 238–247.