Advanced deep learning final project

Hahow courses purchase prediction

Github repo

資料科學碩一 李姵徵

資工 AI 碩一 柳宇澤

資工 AI 碩一 林聖硯

網媒所碩一 何俞彥

Abstract

In this report, we explore the implementation of various machine learning models for the development of a purchase recommendation system, with a focus on providing relevant suggestions to a particular user based on their transaction records. Specifically, we utilize the Hahow dataset to evaluate the performance of these models in the context of a recommendation system.

1 Introduction

The primary objective of this study is to predict future course and topic purchases made by users using machine learning techniques. To inform the choice of appropriate techniques, we conducted exploratory data analysis on a user-item dataset that included information on user course and topic purchases. Data preprocessing techniques, including one-hot encoding and word embeddings, were applied to extract relevant information from the dataset. We also provide a detailed description of the implementation of matrix factorization, logistic matrix factorization, Alternating Least Squares, LightGCN, multi-class logistic regression, and an ensemble method.

The report is structured as follows. In Section 2, we review related work in the area of purchase recommendation systems. Section 3 presents our data and the results of exploratory data analysis. In Section 4, we describe our data preprocessing methods. The implementation of our chosen machine learning techniques is outlined in detail in Section 5. The experimental setup and evaluation of the performance of these methods are described in Section 6. Finally, we conclude in Section 7 and 8.

2 Related work

2.1 Purchase recommendation in machine learning

There are several machine learning models that can be applied in the development of a purchase recommendation system [2]. Some popular options include collaborative filtering models, matrix factorization models, and deep neural networks [3].

Collaborative filtering models, such as those based on the k-nearest neighbors algorithm [16], generate recommendations based on the past behavior of users and items. For instance, if two users have a shared history of purchasing similar items, the model may recommend items that one of the users has purchased but the other has not yet seen.Matrix factorization models, such as those utilizing singular value decomposition (SVD) or non-negative matrix factorization (NMF) [10, 12], represent users and items as low-dimensional vectors and utilize these vectors to make recommendations.Deep neural networks, on the other hand, are capable of learning complex patterns in the data and providing more personalized recommendations [16]. For example, a deep neural network may learn to recommend items to a user based on their previous purchases, the category of items they tend to purchase, and the time of year.

It is worth noting that the selection of an appropriate model for building a recommendation system depends on the specific requirements and characteristics of the problem at hand [1].

2.2 Deep Structured Semantic Models

Deep Structured Semantic Models (DSSM) are a class of neural network models that are designed to learn the latent semantic structure of a given input text or sequence of words. They are typically used in natural language processing (NLP) tasks such as information retrieval, document classification, and machine translation, where the goal is to capture the underlying meaning and relationships between words in a text.

DSSM models are typically trained on a large dataset of text documents, where each document is represented as a sequence of words or tokens. The model learns to map each word or token in the input text to a continuous, low-dimensional vector representation, known as a word embedding. These word embeddings capture the semantic relationships between words, so that words with similar meanings have similar representations.

DSSM models have been widely used in recommendation systems due to

their ability to represent items and users as high-dimensional vectors, which can then be used to compute the similarity between them. This allows recommendation systems based on DSSM to make personalized recommendations based on the interests or preferences of the user.

One of the earliest applications of DSSM in recommendation systems is the work by [18] in which they applied DSSM to collaborative filtering for implicit feedback datasets. They represented items and users as vectors, and used the similarity between the vectors to make personalized recommendations for users.

DSSM has also been used in recommendation systems based on explicit feedback, such as ratings. For example, in the work by [14], the authors used DSSM to represent users and items as vectors, and applied a neural network to make recommendations based on the user's ratings of the items. They demonstrated that their approach was effective at making personalized recommendations and outperformed traditional collaborative filtering methods.

Another application of DSSM in recommendation systems is content-based recommendation, in which recommendations are made based on the similarity between the content of the items. [15] applied DSSM to a large-scale web search engine and demonstrated that it significantly improved the relevance of the search results.

2.3 Graph-based recommendation system

A knowledge graph-based recommender system is a type of recommendation system that utilizes a knowledge graph to generate recommendations. A knowledge graph is a structured representation of real-world entities and the relationships between them [3]. In the context of a recommender system, the entities could be items to be recommended (such as books, movies, or music), and the relationships could be attributes of the items or interactions between the items and users [4].

To generate recommendations, a knowledge graph-based recommender system first constructs a graph of the items and their relationships [19]. It then uses techniques from graph analysis and machine learning to identify items that are related to a given item, or that are likely to be of interest to a particular user [16]. The system can then use this information to recommend items to the user that are similar to items they have liked in the past, or that are related to items they are currently viewing or interacting with [11].

One advantage of knowledge graph-based recommender systems is that they can incorporate a wide range of information about the items and users,

beyond just their explicit ratings or interactions [5]. This allows the system to make more sophisticated and personalized recommendations [17]. For example, a knowledge graph-based recommender system might recommend a book to a user based not only on the fact that they have rated other books in the same genre highly, but also on the fact that the book is written by an author they have expressed an interest in, or that it covers a topic they have previously studied or worked in [16].

3 Data description and exploratory data analysis

In this study, we conducted exploratory data analysis to inform our selection of modeling and data preprocessing techniques for the task of predicting courses and topics that users will purchase in the future. The number of users in the train/valid/test user-item dataset is presented in Table 1.

	# user	# total courses bought	Average courses bought per user	# total topics bought	Average topics bought per user
Train	59,737	139,608	2.337	234,597	3.927
Valid (seen)	7,748	17,817	2.300	28,500	3.678
Valid (unseen)	11,622	19,618	1.688	31,036	2.670
Test (seen)	7,205	-	-	-	-
Test (unseen)	11,907	-	-	-	-

*If a user is marked as « unseen », it means that it didn’t appear in the training set, but we still have its information in the additional supplementary spreadsheet below.

Table 1: Statistics of hahow dataset

In addition to the user-item dataset, we were also provided with three additional spreadsheets containing information about users, courses, and

topics. The following is a summary of the information contained within each of these datasets.

1. User.csv

This csv file contains information about each user, important columns (features) including:

- gender: a column indicates the gender of each user, category contains “female”, “male”, and “other”
- occupation_titles: a column indicates previous jobs of each user, a user could have several occupations, there are a total of 20 occupations
- interests: a column indicates course interests of each user, a user could have several interests, interests could be divided into main interest and sub-interests, there are 12 main interests and 95 sub-interests
- recreation_names: a column indicates hobbies of each user, a user could have several recreations, there are a total of 31 recreations

2. Course_chapter_item.csv

This csv file contains fine-grained information of each course, including descriptions of chapters in each course, descriptions of items in each chapter, the length of the videos in a course, etc. This spreadsheet contains details of a course that users couldn’t see when deciding which course to buy on the website.

3. Courses.csv

This csv file contains the main information of each course. Useful information including name, teacher information, category of this course, learning objective, required tools, target users, etc. This csv file holds essential information of a course that could be seen easily by users when browsing through different courses on the Hahow website, as displayed in figure1.

Figure1: Partial description of a course on the Hahow website

Thus, we believe that these information largely affect the willingness-to-buy of a user.

4 Data preprocessing

In each model, we used different data preprocessing methods to test our ideas inspired by our exploratory data analysis.

For the user-item dataset, we created two user-item interaction matrices for course and topic respectively. If a user has a purchase record for an item (course/topic), we will record it as 1, otherwise it will be 0. In addition, we use scipy.sparse.csr_matrix to create user-item sparse matrices for the alternative least square model. The advantage is that we only need to store the interactive parts of users and items (courses/topics). In other words, there is no need to store 0, which can greatly reduce the use of memory.

For the user information dataset (user.csv), several methods are utilized to extract user information. First, we applied one-hot encoding on all user features, such as gender, occupation, interest, and so on. Furthermore, we used the Chinese word embedding pre-trained by Facebook FastText [18] to convert a user's interest, occupation, and recreation into a 300-dimensional word vector. If the user has multiple features, we sum these word vectors up.

Besides, We concatenated the text information of each user in all columns and used sentence-BERT [19]] to project the user information into a 768-dimensional vector.

For course (course_chapter_item.csv) and topic (courses.csv) information, Sentence-BERT was also utilized to capture course and topic information. In the course_chapter_item.csv, the “sub_group”, “will_learn” and “target_group” columns were concatenated into a long sentence and then transformed into a 768-dimensional vector vai Sentence-BERT. In courses.csv, only the “sub_group” column was used to transform topic information into topic embedding vectors. Then, we compute the inner product of user and course and that of user and topic respectively in order to capture the degree of interest or match of user to course and match of user to topic.

As for label preprocessing, since one user usually bought several courses / topics, we turn these purchase records into hard labels. That is, if a user bought five courses, we turn this purchasing record into five data points rather than one data point with probability equally distributed in five different courses.

5 Modeling approach

5.1 Naive approaches

The Hottest recommendation strategy involves recommending the same 50 most frequently interacted with items to all users. While this approach lacks granularity, it performs well in terms of basic metrics.

For the User Interest Filtering (UIF) approach, we first identify subgroups of users based on their interests and metadata. Next, we combine all items within these subgroups and sort them by popularity, similar to the Hottest approach. Finally, we recommend the top 50 items to users. This method performs better than the Hottest approach.

Additionally, we also experimented with preserving the top 10 UIF items and randomly selecting an additional 40 items for recommendation. We found that this did not significantly decrease the score compared to the original UIF approach, indicating that the top 10 most popular items within each subgroup are widely purchased.

5.2 Matrix factorization and logistic matrix factorization

Matrix factorization (MF) parses the user-item interaction matrix into two low-dimensional matrix products. The prediction purpose is achieved by calculating the elements in the unknown interaction matrix from the known elements in the two low-dimensional matrices.

Logistic matrix factorization (LMF) is based on MF. However, it can capture more implicit information. For example, users have not purchased but had click records or browsing records, and LMF can obtain useful information from these records.

5.3 Alternative least square

The goal of Alternating Least Squares is to find two matrices U and V, such that their product is approximately equal to the interaction matrix R. The input of ALS is two sparse matrices, one for users, and another for items. The sparse matrix can save the memory by only saving the position that has value. ALS is an optimized algorithm that minimizes two loss functions alternatively. It first holds the user matrix fixed and runs gradient descent with the item matrix, then it holds the item matrix fixed and runs gradient descent with the user matrix. The predicted result of one specific user is obtained by the product of the user vector and item vectors.

5.4 LightGCN

LightGCN is a simplified version of the Graph Convolution Network model that removes the linear transformation and activation function. This modification allows LightGCN to operate with high efficiency while maintaining good performance compared to traditional GCN models.

In our task, we could obtain two interaction social networks: the User-Course Network and the User-Topic Network via lightGCN. Both of these networks are bipartite graphs comprising of users and items (courses/topics). LightGCN was used to capture the representations of users and items in each network.

It is worth noting that there is a significant difference in the density of interactions between the two networks. The User-Topic Network has a density that is 13 times that of the User-Course Network. As a result, we used a 3-layered LightGCN on the User-Topic Network and a 5-layered LightGCN on the User-Course Network in order to incorporate more information from interactions.

Finally, we applied Sentence Bert to encode text information as initial embeddings for users and items. This step helps to improve recommendations for unseen users as it preserves the relative semantic meaning in the vector space for unmodified users and items

5.5 Multi-class logistic regression

Multi-class logistic regression is a statistical technique used to predict the outcome of a categorical dependent variable with more than two categories. The aim of logistic regression is to predict the probability that an instance belongs to a particular class, given the values of the independent variables.

This model is optimized using gradient descent and the user specifies the goal and error functions. Various learning rates can be specified to improve performance, and the optimal model is selected through validation.

We utilized the logistic regression module in Sklearn, where the hyperparameter "C" was tuned. C corresponds to the inverse of the regularization strength, with smaller values indicating stronger regularization

5.6 Ensemble

Our ensemble method involves calculating the weighted average of the prediction probabilities (the probability that a user will purchase an item) from multiple models.

For topics, we use ALS, LightGCN, and multi-class logistic regression to calculate the weighted average probabilities. Since ALS performs better on the public leaderboard, we give it a higher weight. The remaining weights are equally divided among the other two models.

For courses, we use LightGCN and multi-class logistic regression to calculate the weighted average probabilities. In this case, multi-class logistic regression performs better on the public leaderboard, so we assign it a slightly higher weight than half. The remaining weight is given to LightGCN.

Table 2 summarizes our models and their corresponding data preprocessing methods.

6 Experiment results

Table 3 and table 4 are the model performance combining our different models and different data preprocessing methods. Best scores are in red bold.

		Data preprocessing methods		
		User-item	Text info.	Text info. + user-item interaction
Model	MF & LMF	User-item matrix	-	-
	ALS	User-item sparse matrix	-	-
	LightGCN	User-item matrix	Sparse matrix + S-bert initial user/item embedding	
	Logistic regression	One-hot encoding	FastText	FastText + S-bert user-course/topic interaction

Table2: Summarization of our models and their corresponding data preprocessing methods

Metric: seen / unseen $\frac{(\text{public mapk@50} + \text{private mapk@50})}{2}$

Models / Features	user-item	text info.	user-item + text info.
Hottest	0.2136 / 0.1717	-	-
MF	0.0278 / 0.0231	-	-
ALS	0.0799 / -	-	-
LightGCN	0.2656 / 0.0970	-	0.2289 / 0.2369
Logistic Regression	0.0111 / 0.0130	0.2702 / 0.2733	0.2764 / **0.3144**
Ensemble	**0.2774** / 0.2370	-	-

*Ensemble weights

Seen topic: LightGCN*0.4 + Logistic Regression*0.6

Unseen topic: LightGCN*0.5 + Logistic Regression*0.5

Table3: Performance of different models in topic prediction tasks

Metric: seen / unseen $\frac{(\text{public mapk@50} + \text{private mapk@50})}{2}$

Models / Features	user-item	text info.	user-item + text info.
Hottest	0.0367 / 0.0501	-	-
UIF	0.0367 / 0.0729	-	-
MF	0.0376 / 0.0271	-	-
LMF	0.0130		
ALS	0.0486	-	-
LightGCN	0.0390 / 0.0172	-	0.0372 / 0.0670
Logistic Regression	0.0390 / 0.0170	0.0466 / 0.0670	0.0469 / 0.0928
Ensemble	**0.0599** / **0.0930**	-	-

*Ensemble weights
Seen course: LightGCN*0.025 + ALS*0.95 + Logistic Regression*0.025
Unseen course: LightGCN*0 + ALS*0.8 + Logistic Regression*0.2

Table4: Performance of different models in course prediction tasks

7 Discussion

Naive methods, which use metadata directly, can achieve acceptable performance, but they lack granularity for personal recommendations. Despite this, they can still serve as a baseline for comparison with other models. It is interesting to note that if we reco mmend the 10 hottest items and randomly select 40 additional items, the performance is similar to recommending the 50 hottest items.

Traditional statistical methods tend to perform poorly when using matrix factorization (MF) due to the need for a dense interaction matrix. In contrast, the logistic regression model performs well on all recommendation tasks, even when using only user text information as model features. This suggests that text information is a meaningful and useful feature in this task. We also found that user and course/topic interactions are useful in predicting future purchases. Incorporating this information into the model significantly improves performance, particularly for predicting seen user preferences.

For deep learning methods, we have tried LightGCN. We find that using deeper aggregation layers improves performance for the user-course dataset since it is much more sparser than the user-topic dataset. In addition, initializing user and item embedding with sentence-bert vector helps with recommendations for **unseen (cold-start) users** (unseen topic: 0.0970→ 0.2369, unseen course: 0.0170→0.0928). However, we found that the initialized user/item embedding created by BERT may be embroiled during the update of weights when running lightGCN algorithm for seen users (seen topic: 0.2656 → 0.2289).

In addition, we attempted to use an ensemble method by adjusting the weights of our regression model, ALS, and LightGCN and combining their outputs. This approach was successful in improving our highest score in each task.

8 Conclusion and future work

The Hahow dataset is an industrially relevant yet processable dataset. As it is derived directly from customer preferences, the information it contains is often hidden within complex raw data. As a result, preprocessing is an important consideration when working with this dataset. Although it may contain some noise, particularly in its text features, the Hahow dataset is still suitable for use in modeling due to its manageable size while larger datasets can be difficult to load onto CPU/GPU for processing.

Through a series of experiments, we found that deep neural network (DNN) models did not perform as well as traditional statistical models, such as logistic

regression, on the Hahow dataset. It is important to carefully consider the use of deep models, as they can require a significant amount of time and resources without necessarily providing improved performance.
We still haven't fully explored the features of this dataset, for example, we didn't take more fine-grained features (chapter content, chapter name, item content, item name, etc) into account. If we have more time to finish this project, we will try to combine these features in our previous models, which may lead to better model performance.

However, we still observed some valuable characteristics of the Hahow dataset that may inform the development of a better recommendation strategy. We hope that these insights could have practical applications in industry in the future.

9 Work distribution

Team member	Experiment	Report
李姵徵	One-hot logistic regression MF Ensemble	Data preprocessing Experiments
柳宇澤	Naïve methods LightGCN Sentence BERT	Discussion Conclusion and future work
林聖硯	FastText User-course/topic interaction Logistic regression	Related work Data description and exploratory data analysis Final check
何俞彥	ALS DeepCTR	Abstract Introduction

Reference

[1] Wei, Kangning, Jinghua Huang, and Shaohong Fu. "A survey of e-commerce recommender systems." 2007 international conference on service

systems and service management. IEEE, 2007.
[2] Desrosiers, Christian, and George Karypis. "A comprehensive survey of neighborhood-based recommendation methods." Recommender systems handbook (2011): 107-144.
[3] Bollacker, Kurt, et al. "Freebase: a collaboratively created graph database for structuring human knowledge." Proceedings of the 2008 ACM SIGMOD international conference on Management of data. 2008.
[4] Zhang, Fuzheng, et al. "Collaborative knowledge base embedding for recommender systems." Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 2016.
[5] Sang, Lei, et al. "Knowledge graph enhanced neural collaborative recommendation." Expert Systems with Applications 164 (2021): 113992.
[6] He, Xiangnan and Deng, Kuan and Wang, Xiang and Li, Yan and Zhang, Yongdong and Wang, Meng. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation, 2020 SIGIR.
[7] Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." Computer 42.8 (2009): 30-37.
[8] Lee, Daniel D., and H. Sebastian Seung. "Learning the parts of objects by non-negative matrix factorization." Nature 401.6755 (1999): 788-791.
[9] Guo, Qingyu, et al. "A survey on knowledge graph-based recommender systems." IEEE Transactions on Knowledge and Data Engineering (2020).
[10] Paatero, Pentti, and Unto Tapper. "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values." Environmetrics 5.2 (1994): 111-126.
[11] Paul Covington, et al. "Deep Neural Networks for YouTube Recommendations." In Proceedings of the 10th ACM Conference on Recommender Systems, 2016.
[12] Huang, Po-Sen, et al. "Learning deep structured semantic models for web search using clickthrough data." Proceedings of the 22nd ACM international conference on Information & Knowledge Management. 2013.
[13] Wang, Xiang, et al. "Kgat: Knowledge graph attention network for recommendation." Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. 2019.
[14] Wang, Xiang, et al. "Learning intents behind interactions with knowledge graph for recommendation." Proceedings of the Web Conference 2021. 2021.
[15] Hu, Yifan, Yehuda Koren, and Chris Volinsky. "Collaborative filtering for implicit feedback datasets." 2008 Eighth IEEE international conference on data mining. Ieee, 2008.

[16] Catherine, Rose, and William Cohen. "Personalized recommendations using knowledge graphs: A probabilistic logic programming approach." Proceedings of the 10th ACM conference on recommender systems. 2016.
[17] Zhang, Shuai, et al. "Deep learning based recommender system: A survey and new perspectives." ACM Computing Surveys (CSUR) 52.1 (2019): 1-38.
[18] Joulin, Armand, et al. "Fasttext. zip: Compressing text classification models." arXiv preprint arXiv:1612.03651 (2016).
[19] Reimers, Nils, and Iryna Gurevych. "Sentence-bert: Sentence embeddings using siamese bert-networks." arXiv preprint arXiv:1908.10084 (2019).