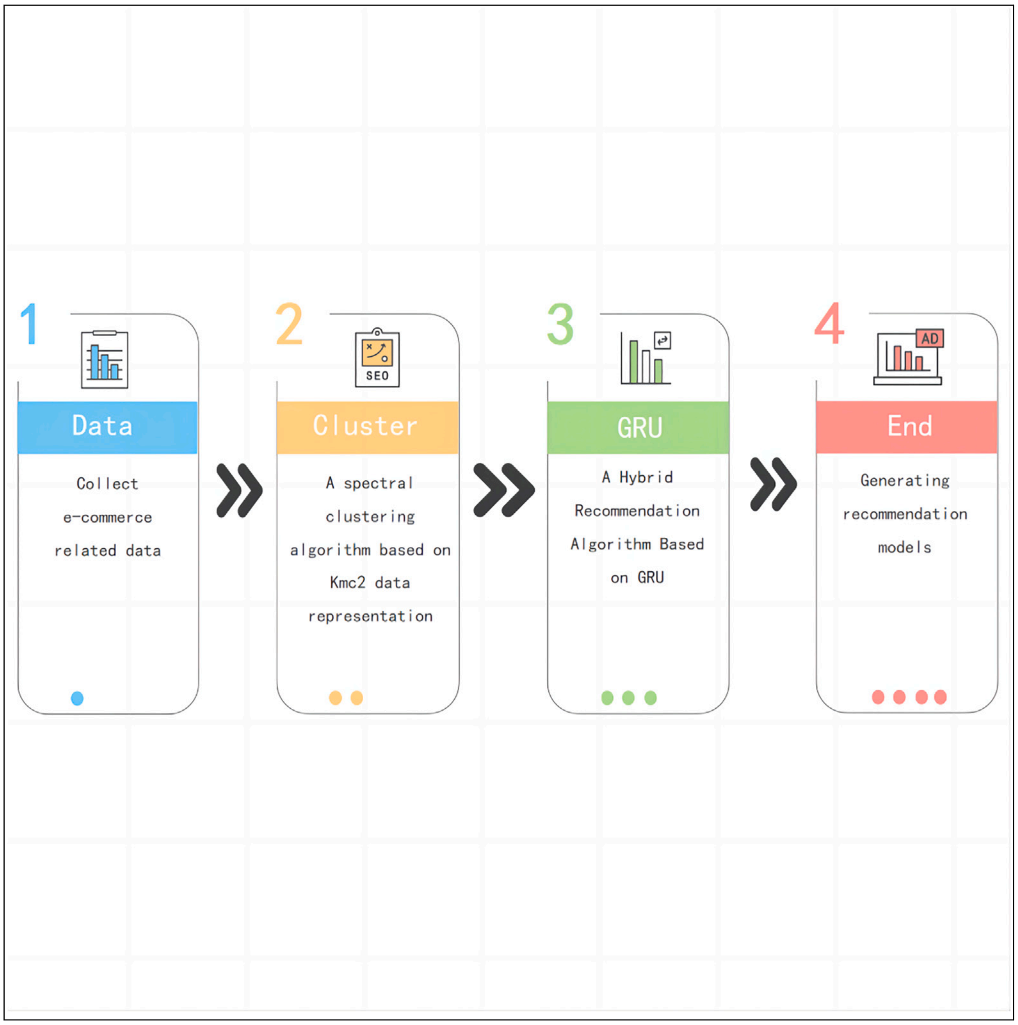
Article

A study on a recommendation algorithm based on spectral clustering and GRU


1
Data
Collect e-commerce related data
2
SEO
Cluster
A spectral clustering algorithm based on Kmc2 data representation
3
GRU
A Hybrid Recommendation Algorithm Based on GRU
4
AD
End
Generating recommendation models

Qingyuan Liu, Ming Yu, Miaoyuan Bai

yuming@nefu.edu.cn (M.Y.)
1038128@qq.com (M.B.)

Highlights

In this paper, propose a recommendation optimization method namely the GRU-KSC algorithm

In this paper, KSC is used to analyze data and generate coding vectors

The HGRU model captures users' dynamic interests and changes in item popularity

Experiments on real datasets

Liu et al., iScience *27*, 108660
February 16, 2024 © 2023 The Author(s).
https://doi.org/10.1016/j.isci.2023.108660

Article

A study on a recommendation algorithm based on spectral clustering and GRU

Qingyuan Liu,[1] Ming Yu,[1,2,*] and Miaoyuan Bai[1,*]

SUMMARY

With the development of e-commerce, the importance of recommendation algorithms has significantly increased. However, traditional recommendation systems struggle to address issues such as data sparsity and cold start. This article proposes an optimization method for a recommendation system based on spectral clustering (SC) and gated recurrent unit (GRU), named the GRU-KSC algorithm. Firstly, this paper improves the original spectral clustering algorithm by introducing Kmc2, proposing a novel spectral clustering recommendation algorithm (K-means++ SC, KSC) based on the existing SC algorithm. Secondly, building upon the original GRU model, the paper presents a hybrid recommendation algorithm (Hybrid GRU, HGRU) capable of capturing long-term user interests for a more personalized recommendation. Experiments conducted on real datasets demonstrate that our method outperforms existing benchmark methods in terms of accuracy and robustness.

INTRODUCTION

With the rapid development of the internet and e-commerce, personalized recommendation systems have become a core component of online platform.[1] These systems analyze user behavior and interests to provide them with tailored product or content recommendations, thereby enhancing user experience, increasing user engagement, and boosting sales and content consumption.[2] However, when confronted with large-scale, high-dimensional, and sparse user behavior data, recommendation systems also face the following challenges.[3]

- Data sparsity: In large-scale recommendation systems, the interaction data between users and items is often very sparse, meaning that many users have no interaction records with many items. This makes it challenging for traditional recommendation algorithms to accurately capture user interests and item characteristics.[4]
- Cold start problem: When new users join the system or new items are introduced, traditional recommendation algorithms struggle to provide accurate personalized recommendations due to the lack of user behavior data. This is known as the cold start problem.[5]
- Lack of deep understanding of users and items: Traditional recommendation algorithms often rely solely on users' historical behavior for recommendations, lacking a deeper understanding of users and items. This can result in less accurate and less personalized recommendations.[6]

To address the aforementioned issues, this article proposes a recommendation system optimization method that combines spectral clustering (SC) and gated recurrent unit (GRU) algorithms.[7] This algorithm can calculate the similarity between users and items using graph theory, providing more comprehensive similarity information. It can also use similarity calculations between users and items to offer similar users and items as references for new users and new item.[8] The algorithm proposed in this article has the capability to handle complex data and contextual information, allowing for a deeper understanding of users and providing more accurate and personalized recommendations. In summary, the algorithm proposed in this article can effectively address issues such as data sparsity, cold start problems, and the lack of deep understanding of users and items.

This article first enhances the original SC and GRU algorithms. It utilizes the Kmc2 data representation, which makes the selection of initial cluster centers in Kmc2 more intelligent and helps prevent the K-means algorithm from getting stuck in local minima. This enhances the stability of the algorithm and reduces sensitivity to initial values. Moreover, due to the intelligent selection of initial centers, Kmc2 often converges to reasonable clustering results more quickly. This reduces the number of algorithm iterations and improves efficiency.

This article utilizes a two-layer GRU network for modeling, which is capable of learning more complex temporal patterns and features. This is because it consists of two stacked GRU layers, each of which can capture data representations at different levels.[9] This contributes to enhancing the model's representational capacity, allowing it to better fit complex sequential data. In summary, the contributions of this article can be summarized as follows.

[1]College of Computer and Control Engineering, Northeast Forestry University, Harbin, Heilongjiang Province, China
[2]Lead contact
*Correspondence: yuming@nefu.edu.cn (M.Y.), 1038128@qq.com (M.B.)
https://doi.org/10.1016/j.isci.2023.108660

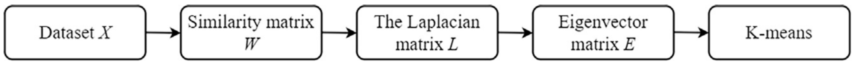


Figure 1. The K-means diagram

- A spectral clustering algorithm based on Kmc2 data representation: The advantages of the spectral clustering algorithm based on Kmc2 data representation in electronic product recommendation systems include its ability to effectively handle large-scale datasets, provide more accurate product recommendations, reduce computational costs, and exhibit good scalability. This algorithm contributes to improving the performance and user experience of recommendation systems
- A hybrid recommendation algorithm based on GRU: The advantages of the dual-layer network model based on GRU in electronic product recommendation systems include its ability to capture both long-term and short-term user interests, thereby improving the accuracy of personalized recommendations. It also possesses strong sequence modeling capabilities, allowing for a better understanding of user behavior sequences and, in turn, better prediction of user needs and preferences. This contributes to enhancing the effectiveness and user satisfaction of electronic product recommendation systems

RESULTS

A spectral clustering algorithm using the Kmc2 data representation

This chapter introduces a spectral clustering algorithm based on Kmc2 data representation. In addition to maintaining the basic process of traditional spectral clustering, this algorithm primarily adds two parts: data representation based on Kmc2 and the reselection of central points. KMC2-spectral clustering is a clustering algorithm that combines Kmc2 data representation with spectral clustering methods. The fundamental idea of this algorithm is to represent data points as Kmc2 feature vectors and then apply spectral clustering techniques to perform cluster analysis of the data. Spectral clustering is a graph-based clustering method that constructs a similarity graph of data and performs spectral decomposition on the graph to achieve clustering. Spectral clustering can, in some cases, handle non-convex clusters and complex data structures better. Therefore, the KMC2-spectral clustering algorithm combines the feature representation of Kmc2 data and the clustering method of spectral clustering. It can be applied to data analysis and clustering tasks, providing more accurate and comprehensive clustering results, especially suitable for dealing with complex datasets.

Near-linear approximation K-means++(Kmc2)

Spectral clustering typically uses a spectral plot (representation of eigenvectors) for clustering, while K-means is a centroid-based clustering method. Spectral clustering can use spectral analysis to determine the eigenvectors of the data and then cluster these eigenvectors using K-means or other clustering algorithms. Therefore, K-means can be used for the final step of spectral clustering, which involves clustering the actual data points based on the eigenvectors.

The performance of spectral clustering relies heavily on the selection of initial cluster centers. Kmc2 (K-means clustering with k-means++ initialization) is a method used to enhance K-means initialization, and it can work well in spectral clustering. Kmc2 uses an improved initialization strategy to select the initial K cluster centers, which helps prevent falling into local optima, thus enhancing the stability and performance of K-means. The k-means diagram is shown in Figure 1.

At the beginning of the algorithm, a random seed center point is selected. The algorithm then iterates through the dataset, constructing a Markov chain in such a way that the stationary distribution of the Markov chain is $p(x)$. The states of the Markov chain after it reaches a stationary state can be considered as samples drawn from $p(x)$. The last (k-1) states are taken as the cluster centers. The target distribution is the objective function, and the meaning of this objective function is that samples that are farther apart have a higher probability of being selected. The proposal distribution is a constant function. The Kmc2 algorithm process is shown in Table 1.

KMC2-spectral clustering

In practical clustering, the sparsity of high-dimensional data and the curse of dimensionality have been significant challenges hindering the development of spectral clustering. To address this issue, this section re-represents the original dataset, completing the subsequent spectral clustering work with the new dataset. The KMC algorithm flowchart is shown in Figure 2.

Given a dataset $X = \{x_1, x_2, \ldots, x_n\}$ in a p-dimensional space, Its mathematical expression is as shown in Equation 1.

$$X = \begin{bmatrix} X_{11} & X_{12} & \ldots & X_{1p} \\ X_{21} & X_{22} & \ldots & X_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ X_{n1} & X_{n2} & \ldots & X_{np} \end{bmatrix} \quad \text{(Equation 1)}$$

Table 1.

Algorithm 1 : Kmc2

```
1 : Input: Dataset X, number of centers k, chain length m
2 : $c_1 \leftarrow$ point uniformly sampled from X
3 : $C_1 \leftarrow c_1$
4 : for(i = 2 to k) do;
5 :     $x \leftarrow$ point uniformly sampled from X
6 :     $dx \leftarrow d(x, C_{i-1})^2$
7 :         if ($dy/dx > Unif(0,1)$) do
8 :             $x \leftarrow y, dx \leftarrow dy$
9 :         end if
10 :    end for
11 :    $C_1 \leftarrow C_{i-1} \cup \{x\}$
12 : end for
13 : return $C_k$
14 : End
```

Since this paper needs to redefine the dataset with the centroids provided by K-means, K-nearest neighbor graph is chosen in this paper.

First, given a dataset *X* and a number of clusters *m* (where $m \ll p$), calculate *m* cluster centroids using the K-means algorithm, and each point still has a dimension of *p*. Its mathematical expression is as shown in Equation 2.

$$C = \begin{bmatrix} X_{11} & X_{12} & \dots & X_{1p} \\ X_{21} & X_{22} & \dots & X_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ X_{m1} & X_{m2} & \dots & X_{mp} \end{bmatrix} \quad \text{(Equation 2)}$$

Step 2, calculate the distance between each point in X and C using the Euclidean distance formula. Its mathematical expression is as shown in Equation 3.

$$dist(i,j) = \sqrt{\sum_{l=1}^{p} (x_{il} - c_{jl})^2} (1 \le i \le n, 1 \le j \le m) \quad \text{(Equation 3)}$$

Through this, we can obtain a distance matrix *Dist* of size *N*M*, Its mathematical expression is as shown in Equation 4.

Step 3: Normalize the obtained matrix using the following formula. Its mathematical expression is as shown in Equations 5–7.

Then we need to use the K-means clustering method to obtain m cluster centroids. This step increases the computational burden of the algorithm. Therefore, this paper has made improvements to the algorithm, and the general process of the algorithm is shown in Table 2.

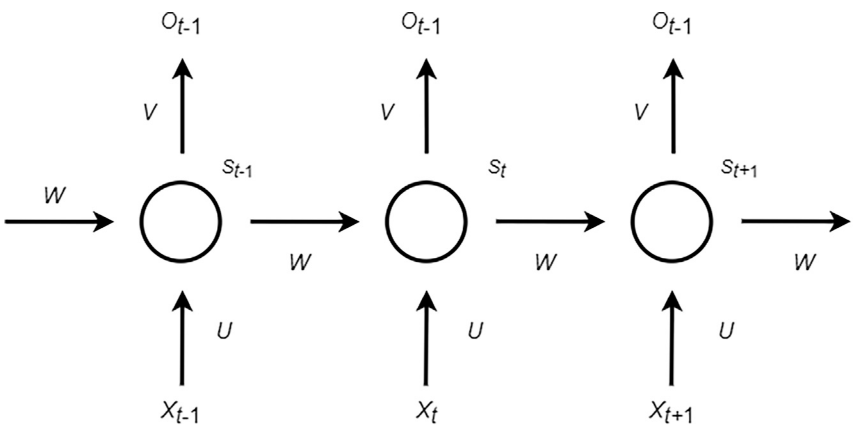


Figure 2. The KMC algorithm

Table 2.

Algorithm 2 : Data representation

```
1 : Input: Dataset X, number of centers m
2 : Return: Dataset Dist'
3 : Begin
4 : $I \leftarrow kmc2, kmc2(m, X)$
5 :     $Dist \leftarrow dist\,(X, I)$
6 :     $Dist' \leftarrow norm(Dist)$
7 : End
```

$$Dist = \begin{bmatrix} d_{11} & d_{12} & \dots & d_{1m} \\ d_{21} & d_{22} & \dots & d_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ d_{n1} & d_{n2} & \dots & d_{nm} \end{bmatrix} \tag{Equation 4}$$

Solve the adjacency matrix and Laplacian matrix for the new dataset, and compute the feature matrix composed of the top *k* eigenvalue eigenvectors. Then, use K-means to cluster the feature matrix to obtain the final clustering result, and the general process of the algorithm is shown in Table 3.

$$Sum_i = \sum_{l=1}^{m} d_{il} \tag{Equation 5}$$

$$d'_{ij} = \frac{d_{ij}}{Sum_i} \tag{Equation 6}$$

$$Dist' = \begin{bmatrix} d'_{11} & d'_{12} & \dots & d'_{1m} \\ d'_{21} & d'_{22} & \dots & d'_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ d'_{n1} & d'_{n2} & \dots & d'_{nm} \end{bmatrix} \tag{Equation 7}$$

Combining the two steps aforementioned, in the entire spectral clustering computation process, this algorithm uses Kmc2 for data pre-processing and final clustering. Therefore, this algorithm is named KSC. KSC does not focus on the construction and eigenvalue decomposition of affinity matrices but rather accelerates spectral clustering with a simple method, making the algorithm more accessible and applicable.

The experimental results are shown in Table 4.

Table 3.

Algorithm 3 : Improved K-means

```
1 : Input: Dataset X, number of centers k, number of iterations iters
2 : Return: k clusters
3 : Begin
4 : $I \leftarrow kmc2, kmc2(k, X)$
5 : while ($iter < iters$) do
6 :     $D \leftarrow dis(X, I)$
7 :     $C \leftarrow D.min()$
8 :     $I \leftarrow$ kmeans ($D$)
9 : end while
10 : return $C_k$
11 : End
```

Table 4. The experimental results

	Taobao user behavior dataset 1/30		Taobao user behavior dataset 1/10		Taobao user behavior dataset 1/3	
Characteristic	Recall	MRR	Recall	MRR	Recall	MRR
GRU-KSC	0.6635	0.2888	0.6817	0.2967	0.5801	0.2467
BPR-MF	0.3216	0.0561	0.1748	0.1029	0.2793	0.1074
S-PoP	0.3341	0.1283	0.2738	0.0944	0.2111	0.1471
GRU-Rec	0.6149	0.2285	0.6035	0.2273	0.4385	0.1536

A GRU-based hybrid recommendation algorithm

To address the personalized recommendation problem in e-commerce, utilizing both long-term user information and short-term sessions, this paper proposes a hybrid gated recurrent unit (HGRU) algorithm based on GRU (gated recurrent unit). GRU is a variant of recurrent neural networks (RNN) with memory capabilities, effectively handling sequential data. This allows the GRU-based hybrid recommendation algorithm to consider user behavior sequences, such as click history and purchase records, better capturing user interests and behavioral evolution.

In the HGRU algorithm, a double-layer GRU network is used for modeling. One layer is used to capture short-term dynamic information of users, while the other layer is used to capture short-term dynamic information of items.

In this paper, the user's session list $[x^u_{i,0}, x^u_{i,1}, x^u_{i,2}, \ldots, x^u_{i,t}]$ is used as input to obtain the hidden state list $[h^u_{i,0}, h^u_{i,1}, h^u_{i,2}, \ldots, h^u_{i,t}]$ for user *i* at time *t*. Here, $x^u_{i,t}$ represents the information of the item clicked by user *i* at time *t*, and $h^u_{i,t}$ represents the hidden state of user *i* at time *t*. Similarly, the hidden state list $[h^g_{j,0}, h^g_{j,1}, h^g_{j,2}, \ldots, h^g_{j,t}]$ for items is computed based on the item input list $[x^g_{j,0}, x^g_{j,1}, x^g_{j,2}, \ldots, x^g_{j,t}]$, where $x^g_{j,t}$ represents user information corresponding to the item *j* clicked at time *t*. The formula for calculating the user's hidden state $h^u_{i,t}$ is as Equation 8.

$$h^u_{i,t} = GRU\left(h^u_{i,t-1}, x^u_{i,t}, c^u_{i,t}\right) \quad \text{(Equation 8)}$$

$c^u_{i,t}$ is the context vector for user *i* at time *t*, and this context vector is computed jointly using the hidden vector $h^u_{i,t-1}$ at time *t*-1 and the input $x^u_{i,t}$ at time *t*. The formula for calculating the hidden state of items at time *t* is as shown in Equation 9.

$$h^g_{j,t} = GRU\left(h^u_{i,t-1}, x^u_{i,t}, c^u_{i,t}\right) \quad \text{(Equation 9)}$$

In this paper, a multilinear interpolation method is used to decode the hidden states of users and items and calculate the correlation value $S_{i,j}$, $S_{i,j}$ between them, as shown in Equation 10.

$$S_{i,j} = h^g_{j,t} W h^u_{i,t} \quad \text{(Equation 10)}$$

In the equation, *W* is a parameter matrix for the multilinear interpolation method, and the number of trainable parameters in this matrix is much smaller than the number of trainable parameters in a fully connected layer. The multilinear interpolation method not only significantly reduces the number of training parameters in the HGRU model but also leads to some improvement in the recommendation performance of the entire hybrid model.

For the initialization of GRU network parameters in the HGRU algorithm, we use the global latent factors K^u_i and K^g_j for users and items as the initialization parameters for the hidden state layer. This initialization method allows the model to train more rapidly compared to using random parameters for initialization.

Finally, I use a softmax layer to normalize the obtained $S_{i,j}$, which normalizes the correlation values of user *i* for various items, resulting in the final score *Score*(*i*, *j*), Its mathematical expression is as shown in Equation 11.

$$Score(i,j) = softmax\left(S_{i,j}\right) \quad \text{(Equation 11)}$$

DISCUSSION

The related work in the field of recommendation algorithms encompasses extensive research, including methods utilizing spectral clustering and GRU models. To set the context and understand the basic concepts in recommendation systems is essential. Recommendation systems aim to provide personalized recommendations to users. There are two main categories of recommendation systems: collaborative filtering and content-based filtering. Collaborative filtering relies on user-item interactions, while content-based filtering uses item features and user profiles.[10]

Spectral clustering has garnered significant attention in recommendation systems. It is a graph-based technique that can uncover the underlying structure in the user-item interaction graph. Researchers have applied spectral clustering to group users or items with similar preferences. This approach aids in identifying user segments or item clusters, enabling personalized recommendations.[11]

GRU is a type of RNN that shows promise in modeling sequential data. In recommendation systems, GRU models are used to capture evolving patterns in user behavior over time. They are particularly effective in modeling both long-term and short-term user interests, making them suitable for sequential recommendation tasks.[12]

Many recent approaches combine various recommendation techniques to leverage the strengths of collaborative and content-based methods. Hybrid systems typically integrate spectral clustering, GRU models, and other algorithms to provide more accurate and diverse recommendations.[13]

Despite advancements in recommendation systems, challenges such as the cold-start problem (for new users or items) and scalability issues for large-scale systems still persist. Researchers continue to explore innovative solutions to address these challenges.[14]

This chapter primarily introduces the spectral clustering algorithm, GRU model used in this paper, and relevant knowledge about recommendation algorithms.

Clustering algorithm

Clustering algorithms are a machine learning technique used to group samples in a dataset into clusters with similar features. Among them, spectral clustering is a graph-based clustering method, with the core idea of representing data samples as nodes in a graph and achieving clustering through spectral analysis of the graph.[15]

The spectral clustering algorithm first represents the samples in the dataset as a graph, typically an undirected graph. In this graph, each data sample corresponds to a node, and the edges between nodes represent the similarity or distance between the samples.[16]

For a graph G, define the set of vertices as V and the set of edges as E, denoted as G(V, E). Where $\vartheta = (\upsilon_1, \upsilon_2, \ldots, \upsilon_n)$, define w_{ji} as the weight between vertex υ_i and υ_j, for an undirected graph, $w_{ij} = w_{ji}$ If there is a connection between two vertices, $w_{ij} > 0$, and if there is no connection, $w_{ij} = 0$. The degree d_i of any vertex υ_i in the graph is the sum of the weights of all edges connected to it. Its mathematical expression is as shown in Equation 12.

$$d_i = \sum_{j=1}^{n} \omega_{i,j} \qquad \text{(Equation 12)}$$

Next, we construct an $N \times N$ degree matrix D using the degrees of all vertices, which is a diagonal matrix. Its mathematical expression is as shown in Equation 13.

$$D = \begin{bmatrix} d_1 & \cdots & \cdots & \cdots \\ \vdots & d_2 & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \cdots & \cdots & \cdots & d_n \end{bmatrix} \qquad \text{(Equation 13)}$$

Using the weight values between all pairs of vertices, we obtain an $N \times N$ adjacency matrix, which is the value corresponding to the ith row and jth column of the adjacency matrix.

In practical applications, it is often not possible to directly provide the weights of the adjacency matrix, and therefore, the adjacency matrix cannot be directly constructed. The fundamental idea in spectral clustering is that the similarity between two points decreases as they are farther apart, and the weight between close points is higher.[16] Therefore, the similarity matrix S can be obtained through the measurement of the similarity of data points to construct the adjacency matrix.[17] Typically, a Gaussian kernel function is used as the measure of similarity between two points.[18] Its mathematical expression is as shown in Equation 14.

The core of the spectral clustering algorithm is to achieve clustering through spectral analysis. To do this, it's necessary to calculate the Laplacian matrix of the graph. There are two common forms of Laplacian matrices: unnormalized and normalized. The normalized Laplacian matrix is typically more commonly used because it takes into account the influence of node degrees, leading to better performance for clusters of different sizes.

$$s_{i,j} = s\left(x_i, y_j\right) = \exp\left(-\frac{\left|\left|x_i - x_j\right|\right|^2}{2\sigma^2}\right) \qquad \text{(Equation 14)}$$

The advantages of the spectral clustering algorithm include its ability to handle non-convex clusters, some robustness to noise and outliers, and applicability to various types of data. The spectral clustering flowchart is shown in Figure 3.

RNN neural network

The basic architecture of an RNN model is relatively simple. At time step t, it receives input x_t, the hidden layer value is s_t, and the final output value is o_t.Its mathematical expression is as shown in Equations 15 and 16.[19]

$$s_t = f(Ux_t + s_{t-1}) \qquad \text{(Equation 15)}$$

$$O_t = g(V_{s_t}) \qquad \text{(Equation 16)}$$

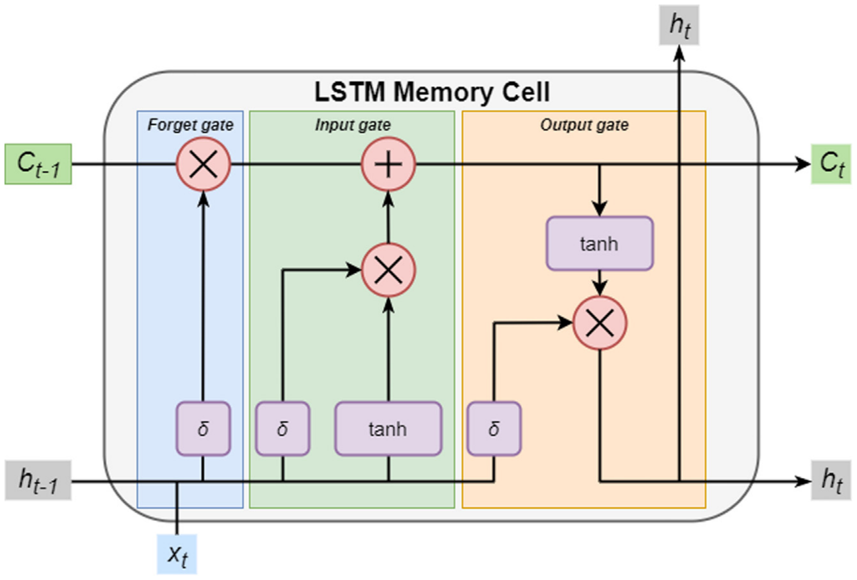


Figure 3. Spectral clustering flowchart

The value of s_t is determined by both x_t and s_{t-1}, where s_{t-1} is the value of the hidden layer at time step t-1, f is the activation function, and U and W are weight matrices.[20]

V is the weight matrix, g is the activation function. Finally, the formula can be obtained as follows Equation 17.[21]

$$O_t = g(Vf(Ux_t + Wf(Ux_{t-1} + Wf(Ux_{t-2} + Wf(Ux_{t-3} + \ldots))))) \qquad \text{(Equation 17)}$$

The RNN model diagram is shown in Figure 4[22]

LSTM (long short-term memory) is a variant of RNN designed to address the issues of vanishing or exploding gradients in the training of long sequences in traditional RNNs. It achieves this by introducing gate mechanisms such as input gate, forget gate, and output gate to control the flow of information, effectively capturing long-term dependencies. This enables LSTM to perform better when dealing with long sequence data, such as text generation and machine translation in natural language processing tasks.[23] The LSTM model diagram is shown in Figure 5[24]

Unlike the LSTM model, the GRU model at time step t needs to maintain not only the hidden state h_t but also a reset gate r_t and an update gate z_t. The specific calculation steps involve first computing the reset gate r_t and update gate z_t, and The GRU model diagram is shown in Figure 6[25]

Recommendation algorithm

Recommender algorithms are a category of computer science and machine learning techniques used to help users discover items or content they might be interested in. In the context of electronic product recommendations, the goal of recommender algorithms is to suggest relevant electronic products to users based on their historical behavior, preferences, and characteristics. Here are some common electronic product recommendation algorithms and methods.[26]

Collaborative filtering

This is a recommendation method based on users' historical behavior. It utilizes user-item interaction data, such as purchase history, ratings, and click records, to identify users with similar interests or items with similar popularity.[27]

Content-based filtering

This approach takes into account the attributes and features of the items themselves, as well as the user's personal preferences. It relies on feature extraction and modeling of items, such as product descriptions, categories, tags, and so on. It then matches the user's preferences with the features of items to generate recommendations.[28]

Matrix factorization

Matrix factorization methods decompose the user-item rating matrix into low-dimensional user and item feature vectors to predict user ratings or preferences for unknown items. Common matrix factorization methods include singular value decomposition (SVD) and latent factor models.[29]

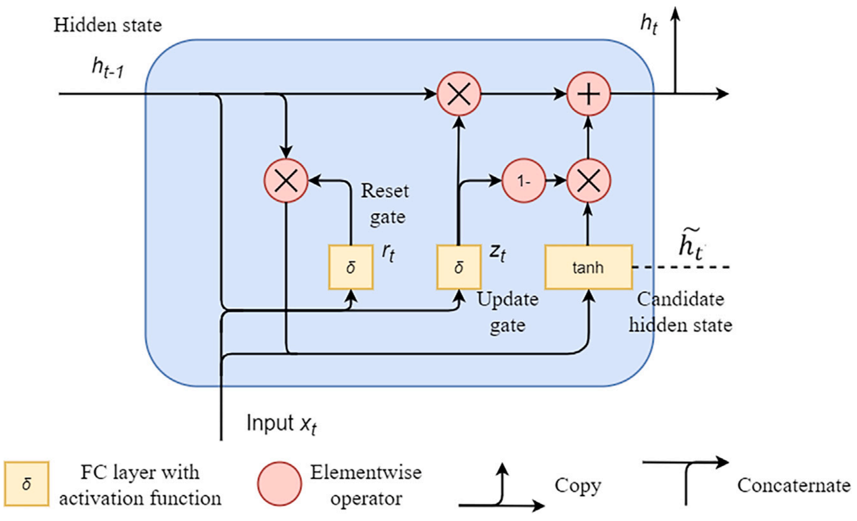


Figure 4. The RNN model diagram

Deep learning models

In recent years, deep learning models such as neural networks have made significant advancements in electronic product recommendations. Multilayer perceptrons (MLP), convolutional neural networks (CNN), and recurrent neural networks (RNN) can be used to learn complex relationships between users and items, providing more accurate recommendations.[30]

Temporal recommendation algorithms

These algorithms take into account the temporal aspects of user behavior, especially the historical behavior of users toward products. They can use sequence models such as recurrent neural networks (RNN) or long short-term memory networks (LSTM) to model users' time-series behavior, better understanding the evolution of user interests.

Hybrid recommender systems

Hybrid recommender systems combine multiple recommendation methods to leverage their strengths. For example, combining collaborative filtering with content-based methods or traditional algorithms with deep learning models to provide more diverse and accurate recommendation results.

STAR★METHODS

Detailed methods are provided in the online version of this paper and include the following:

- KEY RESOURCES TABLE
  - Lead contact
  - Materials availability
  - Data and code availability
- EXPERIMENTAL MODEL AND STUDY PARTICIPANT DETAILS

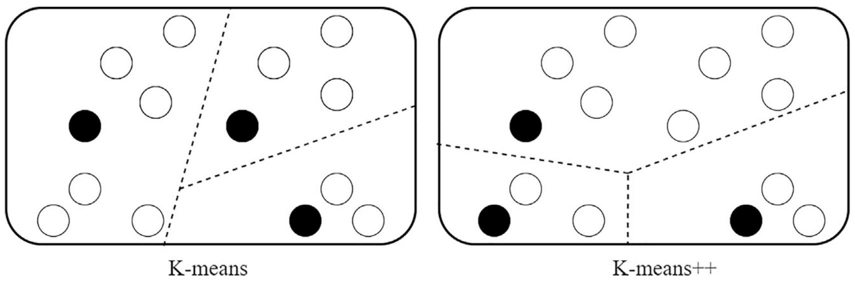


Figure 5. The LSTM model diagram

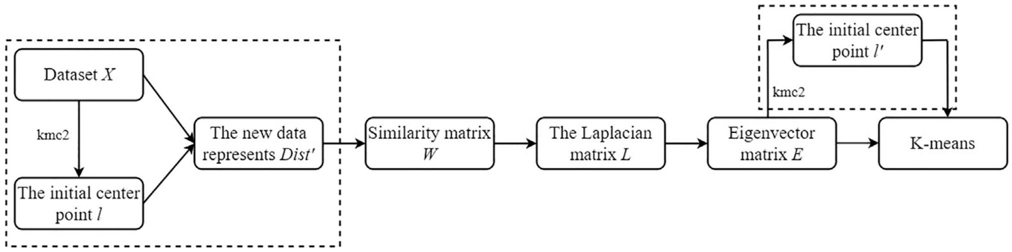


Figure 6. The GRU model diagram

- ○ S-POP
- ○ GRU-Rec
- QUANTIFICATION AND STATISTICAL ANALYSIS

ACKNOWLEDGMENTS

This paper is supported by the Key Research and Development Project of Heilongjiang Province, Research on Big Data Mining and Analysis Techniques for Live E-commerce (2022ZX01A35).

AUTHOR CONTRIBUTIONS

Q.L. and M.Y. conducted the experiments, Q.L. and M.B. designed the experiments and wrote the paper.

DECLARATION OF INTERESTS

The authors declare no competing interests.

Received: October 15, 2023
Revised: November 29, 2023
Accepted: December 5, 2023
Published: December 8, 2023

REFERENCES

1. Hoeppner, F. (1997). Fuzzy shell clustering algorithms in image processing: fuzzy c-rectangular and 2-rectangular shells. IEEE Trans. Fuzzy Syst. *5*, 599–613.
2. Yang, X., Yu, F., and Pedrycz, W. (2020). Typical characteristic-based type-2 fuzzy c-means algorithm. IEEE Trans. Fuzzy Syst. *29*, 1173–1187.
3. Wang, D., Zeng, X.J., and Keane, J.A. (2011). An output-constrained clustering approach for the identification of fuzzy systems and fuzzy granular systems. IEEE Trans. Fuzzy Syst. *19*, 1127–1140.
4. Deng, Z., Jiang, Y., Chung, F.L., Ishibuchi, H., Choi, K.S., and Wang, S. (2016). Transfer prototype-based fuzzy clustering. IEEE Trans. Fuzzy Syst. *24*, 1210–1232.
5. Bai, L., Liang, J., and Guo, Y. (2018). An ensemble clusterer of multiple fuzzy k-means clusterings to recognize arbitrarily shaped clusters. IEEE Trans. Fuzzy Syst. *26*, 3524–3533.
6. Pal, N.R., and Bezdek, J.C. (1995). On cluster validity for the fuzzy c-means model. IEEE Trans. Fuzzy Syst. *3*, 370–379.
7. He, L., Ray, N., Guan, Y., and Zhang, H. (2019). Fast large-scale spectral clustering via explicit feature mapping. IEEE Trans. Cybern. *49*, 1058–1071.
8. Peng, X., Yu, Z., Yi, Z., and Tang, H. (2016). Constructing the l2-graph for robust subspace learning and subspace clustering. IEEE Trans. Cybern. *47*, 1053–1066.
9. Kurth, A., Rnninger, W., Benz, T., Cavalcante, M., Schuiki, F., Zaruba, F., and Benini, L. (2022). An open-source platform for high-performance non-coherent on-chip communication. IEEE Trans. Comput. *8*, 71.
10. Tang, J., Li, C., Zuo, W., Lin, L., and Wang, W. (2016). An approach to streaming video segmentation with suboptimal low-rank decomposition. IEEE Trans. Image Process. *25*, 1947–1960.
11. Ferreira, J.C., Vural, E., and Guillemot, C. (2016). Geometry-aware neighborhood search for learning local models for image superresolution. IEEE Trans. Image Process. *23*, 1354–1367.
12. Abbasi-Sureshjani, S., Favali, M., Citti, G., Sarti, A., and Romeny, B.M.T.H. (2016). Cortically-inspired spectral clustering for connectivity analysis in retinal images: Curvature integration. IEEE Trans. Image Process. *1*, 1.
13. Wu, J., Lin, Z., and Zha, H. (2019). Essential tensor learning for multi-view spectral clustering. IEEE Trans. Image Process. *28*, 5910–5922.
14. Nolfi, J.T. (1999). Learning to perceive the world as articulated: an approach for hierarchical learning in sensory-motor systems. Neural Network. *12*, 1131–1141.
15. Ding, L., Gonzalez-Longatt, Francisco, M., Wall, P., and Terzija, V. (2014). Closure to discussion on "two-step spectral clustering controlled islanding algorithm". IEEE Trans. Power Syst. *29*, 413–414.
16. Rocchetta, R. (2022). Enhancing the resilience of critical infrastructures: Statistical analysis of power grid spectral clustering and post-contingency vulnerability metrics. Renew. Sustain. Energy Rev. *159*, 112185.
17. Apeltsin, L., Morris, J.H., Babbitt, P.C., and Ferrin, T.E. (2010). Improving the quality of protein similarity network clustering algorithms using the network edge weight distribution. Bioinformatics *27*, 326–333.
18. Georgios, A., Pavlopoulos, Charalampos, N., Moschopoulos, Sean, D.,H., and Reinhard. (2009). jclust: a clustering and visualization toolbox. Bioinformatics *25*, 1994–1996.
19. Kirby, N.I., Derose, E.F., London, R.E., and Mueller, G.A. (2004). Nvassign: protein nmr spectral assignment with nmrview. Bioinformatics *20*, 1201–1203.
20. Wu, J., Hu, C., Wang, Y., Hu, X., and Zhu, J. (2020). A hierarchical recurrent neural network for symbolic melody generation. IEEE Trans. Cybern. *6*, 50.
21. Zhang, T., Zheng, W., Cui, Z., Zong, Y., and Li, Y. (2019). Spatial–temporal recurrent neural network for emotion recognition. IEEE Trans. Cybern. *49*, 839–847.

22. Chherawala, Y., Roy, P.P., and Cheriet, M. (2016). Feature set evaluation for offline handwriting recognition systems: Application to the recurrent neural network model. IEEE Trans. Cybern. *46*, 2825–2836.
23. Chi-Hsu, W., and Chun-Yao, C. (2015). Toward a new task assignment and path evolution (tape) for missile defense system (mds) using intelligent adaptive som with recurrent neural networks (rnns). IEEE Trans. Cybern. *45*, 1134–1145.
24. Zhang, P., Xue, J., Lan, C., Zeng, W., Gao, Z., and Zheng, N. (2020). Eleatt-rnn: Adding attentiveness to neurons in recurrent neural networks. IEEE Trans. Image Process. *29*, 1061–1073.
25. Song, X., Jiang, S., Herranz, L., and Chen, C. (2019). Learning effective rgb-d representations for scene recognition. IEEE Trans. Image Process. *28*, 980–993.
26. Zuo, Z., Shuai, B., Gang, W., Liu, X., Wang, X., Wang, B., and Chen, Y. (2016). Learning contextual dependencies with convolutional hierarchical recurrent neural networks. IEEE Trans. Image Process. *25*, 2983–2996.
27. Huang, Z., Chen, H., and Zeng, D. (2004). Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. ACM Trans. Inf. Syst. *22*, 116–142.
28. Zheng, V.W., Zheng, Y., Xie, X., and Yang, Q. (2012). Towards mobile intelligence: Learning from gps history data for collaborative recommendation. Artif. Intell. *184–185*, 17–37.
29. Dick, J., Sloan, I.H., Wang, X., and Woźniakowski, H. (2004). Liberating the weights. J. Complex *20*, 593–623.
30. Liu, J.G., Zhou, T., Wang, B.H., Zhang, Y.C., and Guo, Q. (2009). Effect of user tastes on personalized recommendation. Int. J. Mod. Phys. C *20*, 1925–1932.
31. Elahi, M., Repsys, V., and Ricci, F. (2011). Rating elicitation strategies for collaborative filtering. IEEE J. Sel. Area. Commun. 160–171.

STAR★METHODS

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Software and algorithms		
Python 3.9.6	Python Software Foundation	https://www.python.org
iPython 7.2.6.0	The iPython Development Team	https://www.ipython.org
Raw data	Taobao user shopping behavior data	https://tianchi.aliyun.com/dataset/148605

Lead contact

Further information and requests for resources, reagents should be directed to and will be fulfilled by the lead contact, Ming Yu (yuming@nefu.edu.cn).

Materials availability

This study did not generate new unique reagents.

Data and code availability

This study does not report the original code.

Data reported in this paper will be shared by the lead contact upon request.

Any additional information required to reanalyze the data reported in this paper is available from the lead contact upon request.

EXPERIMENTAL MODEL AND STUDY PARTICIPANT DETAILS

In this section, I will introduce the datasets used for experiments, the experimental environment, several state-of-the-art comparative recommendation algorithms, and the recommendation system evaluation metrics used in this paper. The dataset used for experiments is the Taobao user behavior dataset, and the several state-of-the-art comparative recommendation algorithms include Session-POP, GRU-Rec, BPR-MF, LambdaFM, and others.

Taobao, as one of China's largest e-commerce platforms, has a vast user base and a wide range of products. Therefore, the Taobao user behavior dataset is large-scale and suitable for training recommendation system models with good generalization performance. This dataset is derived from real-world data and reflects the actual behavior of users on the e-commerce platform. Therefore, training recommendation systems using this data can better meet the needs of real-world applications.

To validate the recommendation performance of the HGRU hybrid recommendation algorithm, we will compare HGRU with four other recommendation algorithms that have been proven to be effective. Below, I will provide a brief introduction to these four algorithms.

S-POP

The S-POP algorithm is primarily used to handle user sequence data, such as user behavior sequences (e.g., clicks, views, purchase history) or text sequences (e.g., user comments, search query history). It recommends the next most likely item or action by considering a user's historical behavior.

GRU-Rec

In this paper, we refer to the session-based personalized recommendation algorithm based on GRU networks proposed by Hidasi et al. as GRU-Rec. The algorithm first reduces the input data to the embedding layer and then provides recommendation results through an encoder-decoder approach.[31]

BPR-MF is a type of collaborative filtering recommendation algorithm. Its core idea is to analyze a user's historical behavior data to discover the associations between users and items, and then use these associations for personalized recommendations.

LambdaFM is a recommendation algorithm that combines factorization machines with regularization techniques. It is suitable for various recommendation tasks and has the ability to adjust the smoothness of feature interactions. This algorithm performs well in recommendation systems, particularly for handling data with diverse features.

Mean Reciprocal Rank (MRR) is an internationally recognized metric for evaluating search and recommendation algorithms. If the first result is a match, the score is 1; if n results match, the score is $1/n$; if the clicked item does not appear in the recommendation list, the value is 0. Finally, the scores of all matching results are summed and averaged. A higher MRR indicates that more highly ranked recommended results are clicked by users, making the recommendations better aligned with user needs. MRR is an important measure, particularly in scenarios where the order of recommendations matters. Its mathematical expression is as shown in Equation 18.

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i} \qquad \text{(Equation 18)}$$

In this paper, the four algorithms mentioned above are compared with the algorithm proposed in this paper, with a recommended list length set to 20. The recommendations from these algorithms are evaluated and compared based on two metrics: recall and mean reciprocal rank. Due to the length of the experimental data, the experiments are conducted on 1/30, 1/10, and 1/3 of the data to test the models.

The experimental results are shown in Table 4.

According to the experiments mentioned above, the GRU-KSC algorithm proposed in this paper can achieve better recommendation results.

QUANTIFICATION AND STATISTICAL ANALYSIS

This paper proposes a recommendation system optimization method based on Spectral Clustering (SC) and Gated Recurrent Unit (GRU), namely the GRU-KSC algorithm. The GRU-KSC algorithm consists of two parts: the KSC algorithm and the HGRU model. The algorithm uses the KSC algorithm to analyze data and generate item cluster encoding vectors. Subsequently, the HGRU model captures users' dynamic interests and changes in item popularity. Experiments on real datasets show that our approach outperforms existing benchmark methods in terms of accuracy and robustness.