Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.049780

ARTICLE

Recommendation System Based on Perceptron and Graph Convolution Network

Zuozheng Lian[1,2], **Yongchao Yin**[1] **and Haizhen Wang**[1,2,*]

[1]College of Computer and Control Engineering, Qiqihar University, Qiqihar, 161006, China

[2]Heilongjiang Key Laboratory of Big Data Network Security Detection and Analysis, Qiqihar University, Qiqihar, 161006, China

*Corresponding Author: Haizhen Wang. Email: 01559@qqhru.edu.cn

Received: 14 January 2024 Accepted: 05 March 2024 Published: 20 June 2024

ABSTRACT

The relationship between users and items, which cannot be recovered by traditional techniques, can be extracted by the recommendation algorithm based on the graph convolution network. The current simple linear combination of these algorithms may not be sufficient to extract the complex structure of user interaction data. This paper presents a new approach to address such issues, utilizing the graph convolution network to extract association relations. The proposed approach mainly includes three modules: Embedding layer, forward propagation layer, and score prediction layer. The embedding layer models users and items according to their interaction information and generates initial feature vectors as input for the forward propagation layer. The forward propagation layer designs two parallel graph convolution networks with self-connections, which extract higher-order association relevance from users and items separately by multi-layer graph convolution. Furthermore, the forward propagation layer integrates the attention factor to assign different weights among the hop neighbors of the graph convolution network fusion, capturing more comprehensive association relevance between users and items as input for the score prediction layer. The score prediction layer introduces MLP (multi-layer perceptron) to conduct non-linear feature interaction between users and items, respectively. Finally, the prediction score of users to items is obtained. The recall rate and normalized discounted cumulative gain were used as evaluation indexes. The proposed approach effectively integrates higher-order information in user entries, and experimental analysis demonstrates its superiority over the existing algorithms.

KEYWORDS

Recommendation system; graph convolution network; attention mechanism; multi-layer perceptron

1 Introduction

The Internet has brought great convenience to people, while it has also significantly increased the amount of data, creating information overload. Therefore, people find it challenging to choose the items that best suit their needs. As a significant tool, recommendation systems can address this issue. It performs information filtering, evaluates users' interests based on past activity, and generates customized recommendations [1]. Recommendation systems are being used extensively in e-commerce, social networking, entertainment, education, and other domains. Well-known applications that rely on recommendation systems have emerged, including WeChat, Taobao, Tmall, and so forth.

Deep learning approaches have recently been widely used in natural language, speech, image, and video [2]. Most conventional recommendation system algorithms use deep learning to extract hidden user and item features from the user ID, comment text, and other data and predict users' ratings of the item based on these features to provide suggestions. Following its proposal, researchers discovered that the GCN (graph convolution network) was able to extract user-item connection information that was beyond the reach of conventional recommendation algorithms. Using GCN to create recommendation algorithms has gained increasing popularity in today's recommendation algorithm research.

Presently, most GCN-based recommendation algorithms neglect the unique characteristics of perceptron when extracting user and item features in favor of using the ID of the user's items to capture user features and then using the matrix correlation operation to determine the user's score of related items. Inspired by LightGCN [3], we proposed a novel recommendation method called LGAM (LightGCN Attention Mechanism and MLP) is proposed. The proposed method can more effectively integrate higher-order information from user entries, according to experimental data. It not only achieves a better representation of user characteristics but also significantly enhances the effectiveness of recommendations. The main work of this study is as follows:

(1) The proposed LGAM model designed three main modules: The embedding layer, forward propagation layer, and score prediction layer. The LGAM is built upon the foundation of multi-layer perceptron and GCN. Ultimately, the perceptron, adept at capturing user's interests and preferences, determines the final score.

(2) An attention-integrating GCN is used to implement the forward propagation layer. In order to fully mine the interaction data information between users and items, it can utilize collaborative signals and explicitly encode them in the form of high-order connections by embedded propagation. This reduces both overfitting and data sparsity, better representing user interests and preferences.

(3) Experiments on three standard datasets show that the recall rate and normalized discounted cumulative gain (NDCG) of the proposed LGAM are higher than those of similar algorithms.

2 Related Work

Presently, the techniques for implementing recommendation algorithms mainly include matrix decomposition and deep learning approaches.

2.1 Recommendation Algorithm Basis of Matrix Decomposition

Most conventional recommendation algorithms map users and items to an n-dimensional space using MF (matrix factorization) and then use the mapping to determine the preferences of users for the items—a process known as score prediction. To significantly increase prediction accuracy, recommendation systems, which use MF and biased terms, often use LFM (Latent-Factor Models) [4]. SVD++ (A derivation of the Singular Value Decomposition model) [5] aims to add implicit feedback information based on LFM to correct the prediction results. PMF (Probabilistic Matrix Factorization) model [6] is advanced to improve the missing service score records of the users by updating feature and service feature matrices. Hyper SVD method [7] based on MF using PCA (Principal Component Analysis) for dimensionality reduction is proposed, which is a new method involving the gradual adjustment of the parameters. According to experiment results, the Hyper SVD produces more accurate results than other recommended movies. In order to address the privacy and efficiency challenges faced by MF in collaborative filtering recommendation systems, DS-ADMM (Distributed quantized alternative direction method of multipliers)++ [8] is proposed. DS-ADMM++ integrates differential privacy and utilizes quantized techniques to compress data, which is proven effective by experiments. LSMaOA (Large-Scale Many-Objective Optimization Algorithm) [9] is advanced to improve the MF model for personalized recommendation in the intelligent IoT (Internet of Things). The experimental findings demonstrate the robustness and efficaciousness of LSMaOA in optimizing the six objectives of the model.

2.2 Recommendation Algorithm Basis of Deep Learning

In contrast to deep learning recommendation algorithms, this study investigates GNN (graph neural network), as it can recognize relationships between items and users that deep learning algorithms are unable to. The method basis of neural networks and deep learning is the mainstream direction of recommendation algorithm research. It has been proved that GNN is beneficial for recommendation systems [10]. GCN is one of many GNN implementation techniques that have been widely applied in recommendation systems because of their ability to gradually mine higher-level graph characteristics by adopting multi-layer graph convolution operations [11]. To simulate the high-order relevance of user projects and provide suggestions appropriately, the NGCF (Neural Graph Collaborative Filtering) model [12] is proposed, which introduces GCN into the recommendation algorithm. A-PGNN (A Personalized GNN) [13] is a novel personalized session-aware recommendation approach that primarily combines the Dot-Product attention mechanism and PGNN. Numerous tests demonstrate that A-PGNN performs better than the most advanced related techniques. ExpGCN (Explanation-aware GCN) [14] is advanced and adapted in heterogeneous graph convolutional modeling of recommendation systems, which addresses the limits of computational efficiency and message-passing style. Extensive experiments validate the effectiveness of ExpGCN in explainable recommendations. The attention mechanism is used by the BGANR (Bias-based graph attention neural network recommender) [15] method, which increases the bias term and improves the ability to capture high-order connectivity between nodes. The nia-GCN (Neighbor interaction aware GCN) method [16] takes into account the interaction between neighbor nodes based on GCN, which can effectively collect neighbor node information of all depths. According to the GCN-ONCF (GCN based on Collaborative filtering) model [17], GCN used cross-product operation to convert the coding vector into a two-dimensional feature matrix and a convolution auto-encoder to decompose the convolution matrix. LightGCN, which is based on NGCF, uses a binary graph to reduce the redundant elements of GCN, increasing the model's effectiveness and performance.

3 LGAM Model Design

3.1 Problem Description and Related Definition

As depicted in Fig. 1, the study's core task is to train a model according to all the users u interact with all of the items i to learn the relationship in user u, item i, and besides, according to a multi-layer embedding information user u and item i to further get the user u through the MLP layer and characteristic vector of item i, in the end, get score y_{ui}.

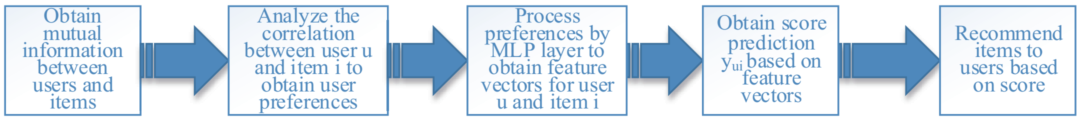

Obtain mutual information between users and items
Analyze the correlation between user u and item i to obtain user preferences
Process preferences by MLP layer to obtain feature vectors for user u and item i
Obtain score prediction y_{ui} based on feature vectors
Recommend items to users based on score

Figure 1: Model's core method

The same method is used to yield the u user's score on all items, and the first K items are recommended to the user u according to the score. The final task is to set the recommendation closer to the u user's future purchase behavior.

3.2 LGAM Model

3.2.1 LGAM Model Overall Introduction

Taking item i_4 and user u_1 as examples, the LGAM (LightGCN Attention Mechanism and MLP) model detailed structure is shown in Fig. 2. The model consists of three main modules: Embedding layer, forward propagation layer, and score prediction layer. Modeling users and objects using interaction data is the role of the embedding layer. In the forward propagation layer, $e_{i_4}^{(0)}$ and $e_{u_1}^{(0)}$ are passed into two parallel GCN with self-connection and weight matrix removed to extract high-order information of user and item, where superscript 0 represents initial embedding. That is the 0 hop expression in the forward propagation layer. Then the final expression of the user and item and e_{i_4} and e_{u_1} is yielded by high-order relation. Lastly, in the score prediction layer, a perceptron extracts the features of the items and users individually. Thus, the prediction score of the user u_1 to i_4 is calculated, and the score of the user u_1 to different items is predicted. According to the prediction score of this set, the first K item contents recommended by the user u_1 are given.

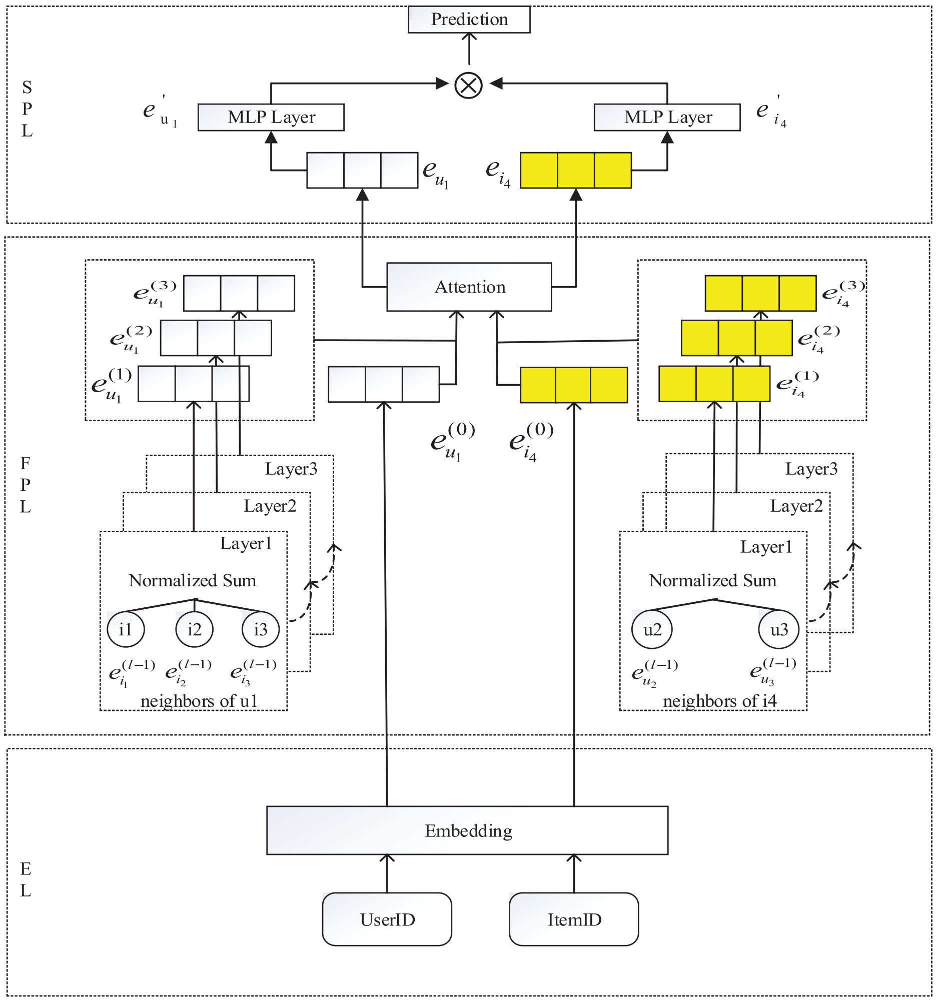


Figure 2: LGAM model structure

3.2.2 Embedded Layer (EL)

The embedding layer is responsible for inputting user and item ID information into the model, and the description of feature embedding is as follows.

Set N users and M items, the ID embedding vector of the ith user is expressed as $e_{u_i}^{(0)} \in R^d$, and the ID embedding vector of the jth item is expressed as $e_{i_j}^{(0)} \in R^d$, where d is the dimension of the embedding vector, which is a regulable hyperparameter. The ID embedding vector of all users forms a set $e_u^{(0)} \in R^{N\times d}$, that is:

$$e_u^{(0)} = [e_{u_1}^{(0)}, \ldots, e_{u_i}^{(0)}, \ldots, e_{u_N}^{(0)}] \tag{1}$$

The ID of all items is embedded into the vector to form the set $e_i^{(0)} \in R^{M\times d}$, that is:

$$e_i^{(0)} = [e_{i_1}^{(0)}, \ldots, e_{i_j}^{(0)}, \ldots, e_{i_M}^{(0)}] \tag{2}$$

The ID embedding vector of both the item and the user is in the initial state, which is further optimized by the forward propagation layer, which enables the ID embedding vector to convey the association relationship it contains more effectively.

3.2.3 Forward Propagation Layer (FPL)

The forward propagation layer is divided into two parallel frameworks to capture the correlation relation between users and items, respectively.

Suppose the dichotomous graph G of all known associations. The method is similar to LightGCN and extracts the association relation between users and items. Taking user u_i and item i_j as examples, the calculation rules of user u_i's ID embedded in GCN network to propagate once (i.e., one-hop) are as follows:

$$e_{u_i}^{(1)} = \sum_{i_j \in N_u} \frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}} e_{i_j}^{(0)} \tag{3}$$

where in $e_{u_i}^{(1)}$ represents the expression of the first hop of the user u_i in GCN network, and $e_{u_i}^{(0)}$ represents the expression of the zero hop of the item i_j, that is the ID embedding of the embedding layer. $1/(\sqrt{|N_u|\,|N_i|})$ follows the aggregation operation in GCN's original design, where N_u represents the set of all neighbor nodes that contain user u_i, and N_i represents the set of all neighbor nodes that contain the item i_j. Eq (3) aggregates the initial ID embedding of all item nodes in all neighbor nodes of the user u_i, thereby yielding the embedding expression of the first hop of the user u_i in graph convolution. Similarly, the calculation rule for embedding the ID of the item i_j in the network is as follows:

$$e_{i_j}^{(1)} = \sum_{u_i \in N_i} \frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}} e_{u_i}^{(0)} \tag{4}$$

First-order propagation (Eqs. (3) and (4)) of nodes in the GCN network models the features of first-order relevance between users and items. The multi-level graph convolution can be stacked in a GCN network to model the features of high-order correlation between users and items by using the computing method of first-order propagation. It can be inferred that the user u_i and item i_j perform convolution operations in the forward propagation layer, and the forward propagation rule for the hop from k to k + 1 is defined as follows:

$$\begin{cases} e_{u_i}^{(k+1)} = \sum_{i_j \in N_u} \frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}} e_{i_j}^{(\mathrm{k})} \\ e_{i_j}^{(k+1)} = \sum_{u_i \in N_i} \frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}} e_{u_i}^{(\mathrm{k})} \end{cases} \tag{5}$$

where $e_{u_i}^{(k+1)}$ and $e_{i_j}^{(k+1)}$ respectively represent the embedded expression of the $(k+1)$ hop of the user u_i and item i_j in graph convolution, $e_{u_i}^{(\mathrm{k})}$ and $e_{i_j}^{(\mathrm{k})}$ respectively represent the embedded expression of the k-hop of the user u_i and item i_j in graph convolution operations.

By using the first-order propagation calculation method, stacking multi-layer graph convolution in the GCN network may be used to model the properties of high-order correlation between users and objects. The problem description is depicted in Fig. 3.

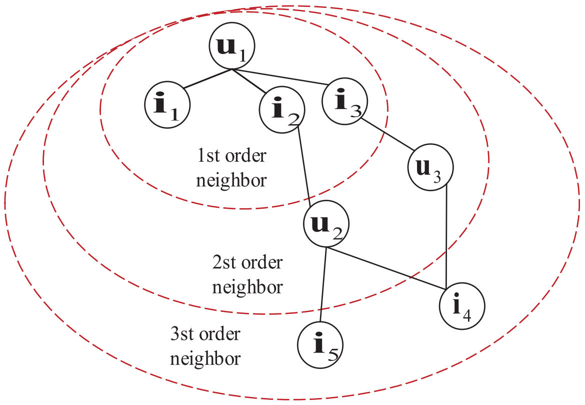


Figure 3: High-level propagation of users and items

User u_1 interacts directly with items i_1, i_2, and i_3; Item i_2 interacts directly with the user u_1 and u_2; Item i_3 directly interacts with the user u_1 and u_3, user u_2 directly interacts with the item i_2, i_4 and i_5, and user u_3 directly interacts with the item i_3 and i_4.

It can be known from Eq. (5), let $\frac{1}{\sqrt{|N_i|}\sqrt{|N_u|}} = \mathrm{P}$, although the N_i and N_u of each node are different, this does not affect the expression of the formula. This step is only for symbolic convenience and has no specific meaning. Therefore, both are set as P and then the target node u_1 is taken as an example to express:

$\mathrm{k} = 0$:

$$e_{u_1}^{(1)} = \mathrm{P}(\mathrm{e}_{i_1}^{(0)} + \mathrm{e}_{i_2}^{(0)} + \mathrm{e}_{i_3}^{(0)}) \tag{6}$$

Eq. (6) represents all the direct neighbor relationships between users and items and the items that the user likes.

$\mathrm{k} = 1$:

$$e_{u_1}^{(2)} = \mathrm{P}(\mathrm{e}_{i_1}^{(1)} + \mathrm{e}_{i_2}^{(1)} + \mathrm{e}_{i_3}^{(1)}) \tag{7}$$

Eq. (7) represents all the second-order neighbor relationships between the users and the items, such as the item that the user likes being liked by other users.

In the expression of Eqs. (6) and (7), $e_{i_1}^{(1)} = Pe_{u_1}^{(0)}$, $e_{i_2}^{(1)} = P\left(e_{u_1}^{(0)} + e_{u_2}^{(0)}\right)$, and $e_{i_3}^{(1)} = P\left(e_{u_1}^{(0)} + e_{u_3}^{(0)}\right)$, since the value of P only affects the coefficient and does not affect the relationship between nodes, it is deleted here. Finally, the statement of $e_{u_1}^{(2)}$ is as follows:

$$e_{u_1}^{(2)} = \mathrm{P}\left(\mathrm{e}_{u_1}^{(0)} + \mathrm{e}_{u_2}^{(0)} + \mathrm{e}_{u_3}^{(0)}\right) \tag{8}$$

k = 2:

$$e_{u_1}^{(3)} = \mathrm{P}\left(\mathrm{e}_{i_1}^{(0)} + \mathrm{e}_{i_2}^{(0)} + \mathrm{e}_{i_3}^{(0)} + \mathrm{e}_{i_4}^{(0)} + \mathrm{e}_{i_5}^{(0)}\right) \tag{9}$$

Eq. (8) represents all the third-order neighbor relationships between the users and the items, such as the popular items among users and friends.

Therefore, each additional layer merges the information of the next layer. Attention factor α^k is introduced to better extract features. Finally, the expression $\hat{e}_{u_i}$ of user u_i and item i_j is obtained, and $\hat{e}_{i_j}$ is:

$$\begin{cases} \hat{e}_{u_i} = \sum_{k=0}^{K} \alpha^k e_u^{(k)} \\ \hat{e}_{i_j} = \sum_{k=0}^{K} \alpha^k e_i^{(k)} \end{cases} \tag{10}$$

where $\alpha^k \geq 0$ represents the importance of kth layer embedding in forming the final embedding, which can be regarded as the model parameter of automatic optimization. In this experiment, the acquisition of α^k using the attentional mechanism results in a satisfactory performance. For user u_i, the attention $\alpha^k \in R^{1\times K}$ of its GCN can be calculated as follows:

$$\alpha_{u_i}^{(k)} = \mathrm{softmax}(\omega_1 \otimes \tanh(\omega_2 \otimes \mathrm{e}_{u_\mathrm{i}}^{(\mathrm{T})})) \tag{11}$$

Among them, the $\omega_1 \in R^{1\times t}$, $\omega_2 \in R^{t\times d}$, t is a super parameter that can be adjusted, $\alpha_{u_i}^k$ contains 0 layer to the first k embedded representation of the weight of $\alpha_{u_i}^{(0)} \sim \alpha_{u_i}^{(k)}$, softmax () function is used to the weight of k layer embedded normalization.

The weighted sum of embedding vectors of each layer by using the attention vector can yield the final embedding representation $e_{u_i} \in R^d$ of the correlation relation of user u_i:

$$e_{u_i} = \sum_{k=0}^{K} \alpha_{u_i}^k e_{u_i}^{(k)} \tag{12}$$

Similarly, the embedding representation $e_{i_j} \in R^d$ of the correlation relation of end-user i_j:

$$e_{i_j} = \sum_{k=0}^{K} \alpha_{i_j}^k e_{ui_j}^{(k)} \tag{13}$$

There are several reasons for the executive-layer composition to achieve the final representation:

(1) The embedding gets excessively smooth as the number of layers rises. Thus, it is challenging to just use the last layer.

(2) Different layers of embedding capture different semantics. As an example, the first layer requires smoothing for users and interactive objects, the second layer smooths users (items) that overlap with users (interactive items), and the higher level records a higher degree of proximity. Therefore, combining them will make the presentation more comprehensive.

(3) The effect of graph convolution and self-connection can be captured by fusing the embeddings of different layers in the form of the weighted sum.

3.2.4 Score Prediction Layer (SPL)

Through the forward propagation layer, the feature vectors $e_{u_i}, e_{i_j} \in R^d$ of the association relation between user u_i and item i_j can be obtained. Then, the vectors e'_{u_i} and e'_{i_j} of user and item can be yielded through the MLP layer, respectively. Finally, the vector of the user u_i is $z_{u_i} \in R^d$, and the calculation of z_{u_i} is shown as follows:

$$\begin{cases} z_1 = \phi_1(e_{u_i}) \\ z_2 = \alpha_2(W_2^T z_1 + b_2) \\ \cdots \\ z_{L-1} = \alpha_L(W_L^T z_{L-2} + b_{L-1}) \\ e'_{u_i} = \alpha_L(W_L^T z_{L-1} + b_L) \end{cases} \tag{14}$$

Similarly, the final form $z_{i_j} \in R^d$ of object i_j is:

$$z_{i_j} = \alpha_L(W_L^T z_{L-1} + b_L) \tag{15}$$

The final score is predicted to be:

$$\hat{y}_{ui} = z_{u_i} \otimes z_{i_j} \tag{16}$$

where ϕ_x denotes the mapping function of the xth layer, W_x^T represents the characteristic transformation matrix of the xth layer, b_x stands for the bias of the xth layer, α_x represents the activation function of the xth layer, σ represents the activation function of the output layer, $\otimes$ represents matrix multiplication, For the activation function of the MLP layer of perceptron, choose sigmoid, hyperbolic tangent (tanh) and Rectifier (ReLU) can also be freely chosen. Each function was analyzed:

1) The sigmoid function constrains each neuron to the range of (0,1), which may limit the performance of the model. It is also prone to overfitting, whereby neurons stop learning when their output approaches zero or one.

2) Tanh is a better choice and has been widely adopted [18,19], but it can only partially alleviate the sigmoid problem because it can be deemed a re-scaled version(tanh(x/2) $= 2\sigma$(x) $-$ 1) of sigmoid.

3) Accordingly, ReLU was chosen as the most biologically reasonable [20]. Furthermore, it promotes sparse activation, which works well with sparse data and reduces the likelihood of overfitting in the model. In conclusion, the empirical results demonstrate that the performance of ReLU is marginally better than that of tanh, while tanh significantly outperforms sigmoid. Finally, the activation function used in this paper is ReLU.

3.2.5 Model Optimization

This paper uses BPR [21] loss, which is often applied in recommendation systems, to learn model parameters and improve the model's ability to express the features of users and items. The foundation of this loss is Bayesian ordering, which considers the relative order of user interactions with objects that are observable and unobservable. It is believed that interactions that are observed have a greater significance than interactions that are not observed. BPR loss is calculated as follows:

$$\text{Loss} = -\sum_{u=1}^{M} \sum_{i\in N_u} \sum_{j\notin N_u} ln\delta\left(\hat{y}_{ui} - \hat{y}_{uj}\right) + \lambda \left\| E^{(0)} \right\|^2 \tag{17}$$

where δ denotes the activation function, the sigmoid() function is used, and λ denotes the adjustable regularization coefficient, which controls L2 regularization. The Adam method [22] is used to optimize it and train it with the mini-batch method $\hat{y}_{ui}$ denotes the predicted score value of the positive sample, and $\hat{y}_{uj}$ denotes the predicted score value of the negative sample.

4 Experimental Analysis

4.1 Preparation

4.1.1 Dataset

To validate the LGAM model, experiments were conducted on three benchmark datasets: Gowalla, Yelp2018, and Amazon-book, which can be open access and differ in domain, size, and sparsity. The statistics for the three datasets are summarized in Table 1.

Table 1: Statistics of the datasets

Dataset	#User	#Items	#Interactions	Density
Gowalla	29,858	40,981	1,027,370	0.00084
Yelp2018	31,048	38,048	1,561,406	0.00130
Amazon-book	52,643	91,599	2,984,108	0.00062

Gowalla: This is the check-in data obtained from Gowalla [23], through which users share their location. To ensure the quality of the data set, 10-core settings were adopted [24], that is, keeping at least 10 interacting users and items.

Yelp2018: This data set was taken from the 2018 version of the Yelp Challenge. Among them, local businesses such as restaurants and bars are considered projects. The same 10-core setup was adopted to ensure data quality.

Amazon-book: Amazon-review is a widely adopted product recommendation data set [25]. Amazon-book was chosen from our collections. Similarly, a 10-core setting was adapted to ensure data quality. For each type of data set, 80% of the historical interaction data of each user was randomly selected as the training set and the rest as the test set. From the training set, 10% of the interactions were randomly selected as the training set to adjust the hyperparameters. For each pair of observed user-item interactions, it was treated as a positive sample. A negative sampling strategy was subsequently performed, pairing it with a negative sample that the user had not previously interacted with.

4.1.2 Evaluation Indicators

For each user in the test set, this article treats all items with which the user does not interact as negative samples. Then, each method outputs the user preference score for all items except the positive sample adopted in the training set. To assess the effectiveness of top-K recommendation and preference ranking, this paper adopted the top-K recommendation method for the recommendation, where K =

20, Recall rate, and NDCG were used to evaluate the model's performance. Recall is used to evaluate the proportion of the number of interactive items used by users in the list of Top20 recommendations to the number of interactive items used by users in the test set. The Recall rate was directly proportional to a better model effect. Assuming that all user sets in the test set are U, for any user $u \in U$, the recommended list of Top20 is L_u, and the real interaction list of user u in the test set is L_u^{test}, the model recall rate calculation is shown in Eq. (18).

$$recall@20 = \sum\nolimits_{u \in U} \frac{1}{|U|} \frac{\left|L_u \bigcap L_u^{test}\right|}{\left|L_u^{test}\right|} \tag{18}$$

The correlation score of the recommendation results at different points in the recommendation list is measured by the NDCG. The higher the ranking, the greater the suggestion effect, and the higher the score of an item, the more relevant it is to the user. Assuming L_u^i is the recommendation for the ith position in the Top20 recommendation list, f(x) is set to 1 when x > 0, and 0 when vice versa. Set the correlation score $rel \in \{0, 1\}$, the NDCG calculation method is shown in Eq. (19).

$$NDCG@20 = \sum\nolimits_{u \in U} \frac{1}{|U|} \frac{DCG_u@20}{IDCG_u@20} = \sum\nolimits_{u \in U} \frac{1}{|U|} \frac{\sum_{i=1}^{20} \frac{f(L_u \bigcap L_u^{test})}{\log_2(i+1)}}{\sum_{i=1}^{20} \frac{f(L_u^{test_i})}{\log_2(i+1)}} \tag{19}$$

4.1.3 *Experimental Environment*

GPU can execute high-intensity matrix operations in parallel, completing more computations in the same amount of time because the proposed model requires a lot of computation. GPU is capable of using bigger datasets, increasing training speed, enhancing model performance, and making better use of computational resources. Therefore, this paper chooses to conduct corresponding experimental training on GPU. Table 2 describes the experimental environment settings.

Table 2: Experimental environment settings

Experimental environment	Settings
Operating system	Windows10
GPU	GTX 2080
Memory	16.0 GB
Programming language	Python
Deep learning framework	PyTorch

4.2 *Compare the Model*

The main comparison method is LightGCN, outperforming a variety of methods, including the graph convolutional matrix completion (GC-MC) model [26], PinSage [27], NeuMF basis of the neural network [28], the collaborative memory network model (CMN) [29], factorization-based models MF and Hop-Rec [30]. Comparisons are made under the same assessment criteria as follows:

(1) Mult-VAE [31] is designed based on item-based Collaborative Filtering (CF) and variational automatic coding (VAE), which supposes that data is produced from polynomial distribution and uses variational inference for parameter evaluation.

(2) NGCF successfully integrates GCN into the recommendation system by adhering to the conventional GCN concept. To forecast scores, it effectively extracts embedded interactions (correlation relationships) between users and objects using binary graphs.

(3) Based on NGCF, LightGCN eliminates redundant feature transformation and activation functions, which significantly improves the efficiency and prediction accuracy of the model.

(4) Graph regularized matrix factorization (GRMF) [32] is a way of smoothing matrix decomposition by adding graph Laplacian regularization.

(5) The LII-GCCF (linear transformation, initial residual, identity mapping, graph convolutional collaborative filtering) model [33] is an improved recommendation algorithm using GCN.

4.3 Hyperparameter Adjustment of Experimental Scheme and Model

The embedding size of all models is fixed at 64, and the embedding parameters are initialized by the Xavier method [34]. LGM was optimized using Adam, the default learning rate was 0.001, and the default mini-batch size was 1024 (for Amazon-book data sets, this study increased the mini-batch size to 2048 to improve speed). The L2 regularization coefficient λ is set to 1e-4, and the layer combination coefficient α^k is obtained using the attention mechanism. Moreover, an early stop policy is implemented; that is, an early stop occurs if there is no increase in *recall*@20 on validated data over 50 consecutive epochs.

This paper adjusted the graph convolution iteration number n in the experimental part. The dimension of the embedding vector (d = 64) and other parameters in the comparative experiment of varying the number of iterations of graph convolution are consistent with those mentioned previously, as Table 3 illustrates. By examining and comparing the experimental data in Table 3, it can be inferred that the model's optimal effect is achieved at n = 3. While certain data sets exhibit superior results at n = 4, this will complicate training and limit its benefits, leading to a final iteration number of 3 layers. Considering the sparsity of the data set, it can be inferred that most nodes in the GCN constructed by the three data sets, respectively, have 2-hop and 3-hop paths, and a few have paths above 3-hop (which can be verified by the data set). In this case, more correlations can be extracted when n = 2 than n = 1, and the model impact will be improved. When n = 3, more correlations are extracted than n = 2, which improves the effect of the model. When n = 4, more correlation can be extracted than n = 3; however, the model effect will not be better but will be relatively reduced.

Table 3: Contrast experiments of the iteration number of GCN

	Dataset					
Layer	Gowalla		Yelp2018		Amazon-book	
n	Recall	NDCG	Recall	NDCG	Recall	NDCG
1	0.1782	0.1508	0.0552	0.0526	0.0401	0.0302
2	0.1792	0.1546	0.0644	0.0542	0.0434	0.0336
3	0.1883	0.1598	0.0706	0.0592	0.0434	0.0357
4	0.1886	0.1587	0.0711	0.0594	0.0430	0.0352

To model the signal of high-order connectivity coding, the depth of high-order relevance in LGAM is set as 3, and the layer number of perceptron MLP is set as 2. This article presents the findings of three embedded propagation layers with a 0.1 message loss rate and a 0.0 node loss rate without any more explanation. Typically, the model can converge after 1000 epochs.

4.4 Performance Comparison

Table 4 demonstrates that LGAM outperforms alternative approaches across all data sets. Its efficiency is evidenced by a straightforward and rational design, where the model's best outcomes are bolded.

Table 4: Performance comparison of seven models on three datasets

	Dataset					
	Gowalla		Yelp2018		Amazon-book	
Method	Recall	NDCG	Recall	NDCG	Recall	NDCG
NGCF	0.1570	0.1327	0.0579	0.0477	0.0344	0.0263
Mult-VAE	0.1641	0.1335	0.0584	0.0450	0.0407	0.0315
GRMF	0.1477	0.1250	0.0571	0.0462	0.0354	0.0270
GRMF-norm	0.1557	0.1261	0.0561	0.0454	0.0352	0.0269
LightGCN	0.1830	0.1554	0.0649	0.0530	0.0411	0.0315
LGAM	**0.1883**	**0.1598**	**0.0656**	**0.0542**	**0.0434**	**0.0327**
LII-GCCF	0.1810	0.1455	0.0640	0.0532	0.0402	0.0315

The performance of GRMF is comparable to that of NGCF. GRMF-norm outperforms GRMF on Gowalla and adds little value on Yelp2018 and Amazon-book by normalizing the Laplacian regularizer. Because of the non-linear interaction between user and object features during feature extraction and the addition of a perceptron, the LGAM model performs marginally better than the LightGCN model.

As can be seen from the above Table 4:

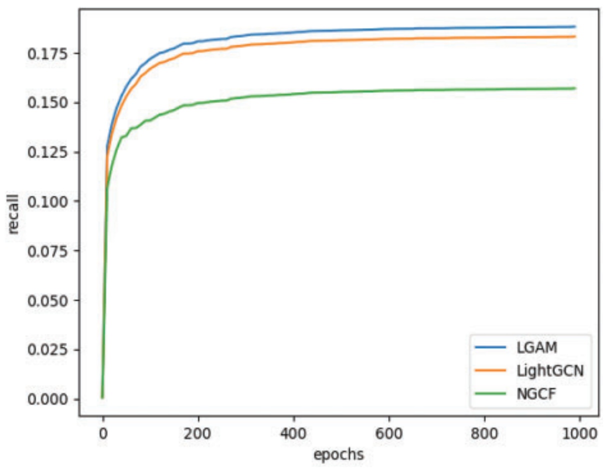
(1) LGAM outperforms slightly than LightGCN, and LightGCN is marginally superior to other methods. For instance, on Gowalla, the highest recall rate recorded in the LightGCN is 0.1830; however, in this study, the NDCG is 2.83% higher, and the LGAM can reach 0.1883 in a 3-layer setting, which is 2.90% higher. For example, the results of LGAM on the Gowalla dataset are shown in Fig. 4.

(2) Raising the number of layers improves performance but reduces the benefits. The usual observation is that raising the number of layers from 0 to 1 results in maximum performance gain and adopting the number of layers three results in satisfactory performance in most cases.

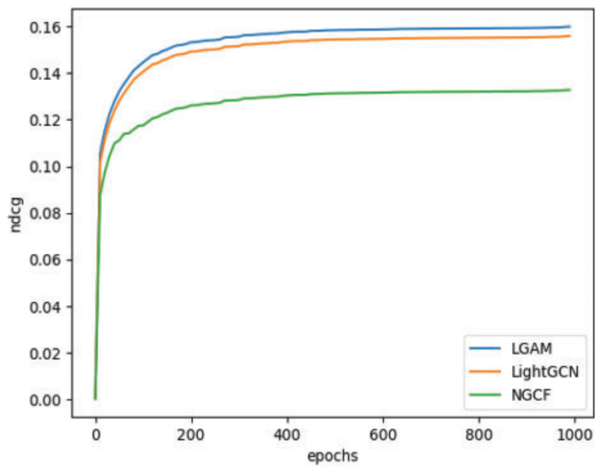
(3) LGAM consistently produced reduced training loss during the training process, effectively translating to improved test accuracy and demonstrating LGAM's potent generalization capacity. Although improving the learning rate of LightGCN can reduce its training loss, it cannot improve the test recall rate because reducing training loss is only a temporary solution for LightGCN.

4.5 Ablation Experiment

Ablation tests were performed to validate the efficacy of the LGAM model further. The hyper-parameter adjustment discussed in Section 4.3 is consistent with the parameters of the LGAM model without an attention mechanism, which we will refer to as LGM. The graph convolution iteration number is 3. The comparison results of the LGM model's recall rate and NDCG on the Gowalla dataset are shown in Fig. 5. Thus, it can be concluded that the attention mechanism has a positive effect on the model. In terms of recall, LGAM has improved by 1.074% compared to LGM, and the normalized cumulative loss gain of this model has increased by 2.04% compared to LGM.

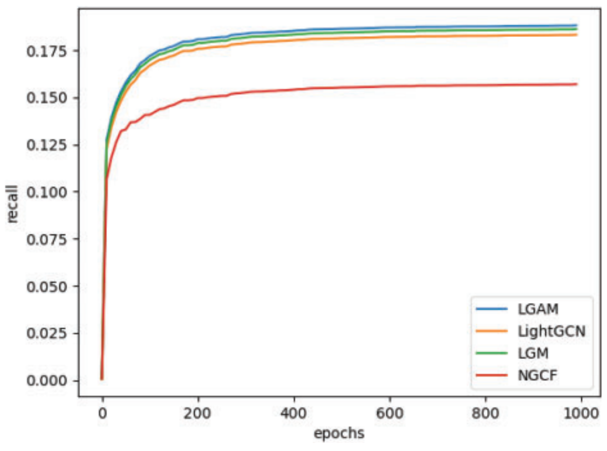


(a) Recall in the Gowalla dataset

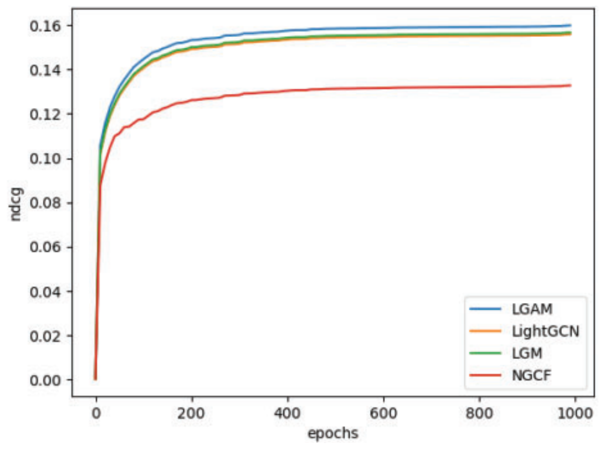


(b) NDCG in the Gowalla dataset

Figure 4: Evaluation metrics on the Gowalla dataset



(a) Recall rate on Gowalla dataset



(b) NDCG on Gowalla dataset

Figure 5: Ablation experiment on the Gowalla dataset

5 Conclusion

To fully extract the features of users' items, this study proposes a hybrid algorithm known as LGAM. The model combines the strengths of graph neural networks and attention mechanism, consisting of closely related modules: EL, FPL, and SPL; notably, the FPL plays a crucial role. It employs two parallel GCNs to simultaneously extract the features from users and items. By stacking multi-layer graph convolutions and incorporating a carefully designed attention factor, it effectively captures higher-order correlations and association relevance information between users and items, better expressing user interests and preferences.

Experimental results on three datasets show that LGAM achieves higher recall and NDCG compared to other methods. This indicates its effectiveness in enhancing recommendation performance. In the future, frameworks such as knowledge graphs and neighborhood aggregation may be explored to optimize the model further. Additionally, the study will delve into the impacts of different feature fusions on the recommendation system.

Acknowledgement: None.

Funding Statement: This work was supported by the Fundamental Research Funds for Higher Education Institutions of Heilongjiang Province (145209126) and the Heilongjiang Province Higher Education Teaching Reform Project under Grant No. SJGY20200770.

Author Contributions: Writing—review and editing, supervision and funding acquisition: Z. Lian; validation and writing—original draft: Y. Yin; conceptualization, methodology, and formal analysis: H. Wang. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data used in the study are all available.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. Yao, "Review of personalized recommendation system," *China Collect. Econ.*, no. 25, pp. 71–72, 2020.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. doi: 10.1038/nature14539.

[3] X. He, K. Deng, X. Wang, and Y. Li, "LightGCN: Simplifying and powering graph convolution network for recommendation," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Dev. Inf. Retriev.*, Xi'an, China, 2020, pp. 639–648.

[4] Y. Deldjoo, T. D. Noia, and F. A. Merra, "A survey on adversarial recommender systems," *ACM Comput. Surv.*, vol. 54, no. 2, pp. 1–38, 2022. doi: 10.1145/3439729.

[5] M. Jallouli, S. Lajmi, and I. Amous, "When contextual information meets recommender systems: Extended SVD++ models," *Int. J. Comput. Appl.*, vol. 44, no. 4, pp. 349–356, 2020. doi: 10.1080/1206212X.2020.1752971.

[6] L. Duan, T. Gao, W. Ni, and W. Wang, "A hybrid intelligent service recommendation by latent semantics and explicit ratings," *Int. J. Intell. Syst.*, vol. 36, no. 2, pp. 7867–7894, 2021. doi: 10.1002/int.22612.

[7] B. Geluvaraj and M. Sundaram, "A Matrix factorization technique using parameter tuning of singular value decomposition for recommender systems," *Turk. J. Comput. Math. Educ.*, vol. 12, no. 2, pp. 3313–3319, 2021.

[8] F. Zhang, E. Xue, R. Guo, G. Qu, G. Zhao and A. Y. Zomaya, "DS-ADMM++: A novel distributed quantized ADMM to speed up differentially private matrix factorization," *IEEE Trans. Parallel Distrib. Syst.: Publ. IEEE Comput. Soc.*, vol. 33, no. 6, pp. 1289–1302, 2022. doi: 10.1109/TPDS.2021.3110104.

[9] B. Cao, Y. Zhang, J. Zhao, X. Liu, and Z. Lv, "Recommendation based on large-scale many-objective optimization for the intelligent internet of things system," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 15030–15038, 2021. doi: 10.1109/JIOT.2021.3104661.

[10] Y. Guo, Y. Ling, and H. Chen, "A time-aware graph neural network for session-based recommendation," *IEEE Access*, vol. 8, pp. 167371–167382, 2020. doi: 10.1109/ACCESS.2020.3023685.

[11] J. Chen, G. Lin, J. Chen, and Y. Wang, "Towards efficient allocation of graph convolutional networks on hybrid computation-in-memory architecture," *Sci. China: Inf. Sci.*, vol. 64, no. 6, pp. 108–121, 2021. doi: 10.1007/s11432-020-3248-y.

[12] X. Wang, X. He, M. Wang, F. Feng, and T. S. Chua, "Neural graph collaborative filtering," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Dev. Inf. Retriev.*, Paris, France, 2019, pp. 165–174.

[13] M. Zhang, S. Wu, M. Gao, X. Jiang, K. Xu and L. Wang, "Personalized graph neural networks with attention mechanism for session-aware recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 8, pp. 3946–3957, 2022. doi: 10.1109/TKDE.2020.3031329.

[14] T. Wei, T. W. S. Chow, J. Ma, and M. Zhao, "ExpGCN: Review-aware graph convolution network for explainable recommendation," *Neural Netw.*, vol. 157, no. 9, pp. 202–215, 2023. doi: 10.1016/j.neunet.2022.10.014.

[15] J. Wang, Q. Wen, X. Yang, and Q. Zhang, "Recommended attention neural networks algorithm based on deviation," *Control Decis.*, vol. 37, no. 7, pp. 1705–1712, 2020.

[16] J. Sun *et al.*, "Neighbor interaction aware graph convolution networks for recommendation," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Dev. Inf. Retriev.*, Xi'an, China, 2020, pp. 1289–1298.

[17] J. Su, T. Xu, X. Zhang, Y. Shi, and S. Gu, "Collaborative filtering recommendation based on convolution and outside product model," *Comput. Appl. Res.*, vol. 38, no. 10, pp. 3044–3048, 2021.

[18] M. A. Mercioni and S. Holban, "Developing novel activation functions in time series anomaly detection with LSTM autoencoder," in *Proc. 2021 IEEE 15th Int. Symp. Appl. Comput. Intell. Inf. (SACI)*, Timişoara, Romania, 2021, pp. 000073–000078.

[19] R. Gnanasambandam, B. Shen, J. Chung, X. Xue, and Z. Kong, "Self-scalable Tanh (Stan): Faster convergence and better generalization in physics-informed neural networks," arXiv:2204.12589, 2022.

[20] L. L.Parisi, D. Neagu, R. Ma, and F. Campeanet, "Quantum ReLU activation for convolutional neural networks to improve diagnosis of Parkinson's disease and COVID-19," *Expert Syst. Appl.*, vol. 187, no. 1, pp. 115892, 2022.

[21] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thiemeet, "BPR: Bayesian personalized ranking from implicit feedback," arXiv:1205.2618, 2012.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv:1412.6980, 2014.

[23] D. Liang, T. McInerney, and D. Terzopoulos, "Modeling user exposure in recommendation," in *Proc. 25th Int. Conf. World Wide Web*, 2016, pp. 951–961.

[24] R. He and J. McAuley, "VBPR: Visual Bayesian personalized ranking from implicit feedback," in *Proc. 30th AAAI Conf. Artif. Intell.*, Phoenix, USA, vol. 30, 2016, pp. 1–7.

[25] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *Proc. 25th Int. Conf. World Wide Web*, arXiv:1602.01585, 2016.

[26] R. Berg, T. Kipf, and M. Welling, "Graph convolutional matrix completion," in *Proc. KDD'18 Deep Learn. Day*, London, UK, 2017, pp. 1–7.

[27] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, London, UK, 2018, pp. 974–983.

[28] X. He, L. Liao, H. Zhang, L. Nie, and T. S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, Perth, Australia, 2017, pp. 173–182.

[29] T. Ebesu, B. Shen, and Y. Fang, "Collaborative memory network for recommendation systems," in *41st Int. ACM SIGIR Conf. Res. Dev. Inf. Retriev.*, Ann Arbor, USA, 2018, pp. 515–524.

[30] J. H. Yang, C. M. Chen, C. J. Wang, and M. F. Tsai, "HOP-rec: High-order proximity for implicit recommendation," in *Proc. 12th ACM Conf. Recommender Syst.*, Vancouver, Canada, 2018, pp. 140–144.

[31] D. Liang, R. G. Krishnan, M. D. M.D.Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *Proc. 2018 World Wide Web Conf.*, Lyon, France, 2018, pp. 689–698.

[32] N. Rao, H. F. Yu, P. K. Ravikumar, and I. S. Dhillon, "Collaborative filtering with graph information: Consistency and scalable methods," *Adv. Neural Inf. Process. Syst.*, vol. 28, pp. 1–9, 2015.
[33] R. Guo, X. Li, Y. Hu, and M. Qu, "A simple graph convolutional network with abundant interaction for collaborative filtering," *IEEE Access*, vol. 9, pp. 77407– 77415, 2021. doi: 10.1109/ACCESS.2021.3083600.
[34] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Stat.*, Sardinia, Italy, 2010, pp. 249–256.