ISSN 2616-5775 Vol. 7, Issue 4: 56-61, DOI: 10.25236/AJCIS.2024.070408

Attention-based Multilayer Linear Diffusion Model

Yixuan Zhang[1,*]

[1]*College of Information Engineering, Nanjing University of Finance and Economics, Nanjing, China*
[*]*Corresponding author: 670891548@qq.com*

Abstract: *In recent years, two major issues in recommendation systems have received extensive attention from researchers: (1) The incompleteness of user interaction data: In recommendation systems, user interaction data is often incomplete. (2) Difficulty in accurately reflecting users' true preferences through interactions: User interaction data may suffer from selection biases, with some interactions possibly being noise, leading to misinterpretation of user preferences by the recommendation system. To address these challenges, we propose an Attention-based Multilayer Linear Diffusion Model (AMLDM). Specifically, we gradually introduce pre-defined Gaussian noise into the forward process to disrupt users' interaction histories. Subsequently, through multiple attention-based linear layers, we iteratively restore the damaged interaction histories incurred during th e forward process, ensuring that the distribution of the repaired interaction history aligns with the original distribution of user interaction history. By injecting an appropriate amount of noise into users' interaction histories, we enhance the robustness of the recommendation system and learn users' true preferences during the reverse process. Comparative analysis with several benchmark recommendation system models demonstrates the significant advantages of our proposed algorithm in recommendation performance.*

Keywords: *Recommendation System, Neural Network, Diffusion Model*

1. Introduction

For modern recommendation systems, leveraging historical interaction data between users and items to learn user preferences and recommend items is crucial. Traditional deep learning approaches, such as Multilayer Perceptrons (MLP), Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), and the Transformer framework, typically represent items as dense vectors in a latent space. However, this approach encounters two main challenges:(1)User interests are diverse and may change over time, making it difficult for fixed-length vectors to fully capture user preferences.(2)These methods assume that items with the highest correlation to user interactions are the most relevant, potentially leading to exposure bias.To address these issues, researchers have employed generative models such as Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) to infer the interaction probabilities of users without item interaction records. These models treat user preferences as latent factors determining user behavior and sample from learned probability distributions to mimic the uncertainty in user behavior.

GAN learns the judging criteria in the discriminator through adversarial training. For instance, in IRGAN (IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models), researchers proposed a generative retrieval GAN model for item recommendation; while SD-GAR (Sampling-Decomposable Generative Adversarial Recommender) further improved this process by sampling decomposition in the generator. However, adversarial training suffers from significant instability, leading to uncontrollable performance of recommendation systems based on GANs.

VAE learns the posterior distribution of latent factors by maximizing the likelihood of historical interactions. For example, Liang et al. proposed VAE for collaborative filtering with implicit feedback; while in ACVAE (Adversarial and Contrastive Variational Autoencoder for Sequential Recommendation), authors utilized adversarial training to enhance sequential VAE for capturing user preferences. Despite VAE having precise mathematical theoretical support, computing the posterior distribution in VAE is very challenging.

In addition to the aforementioned issues, both GANs and VAEs may suffer from posterior collapse due to information bottleneck, where the posterior probability degenerates to be consistent with the prior probability, resulting in hidden representations possibly lacking information about user preferences. As these methods generate only a small number of outputs over multiple iterations, they are also prone to

ISSN 2616-5775 Vol. 7, Issue 4: 56-61, DOI: 10.25236/AJCIS.2024.070408

pattern collapse issues. To address these challenges, researchers have employed a new generative model paradigm—diffusion models, which overcome the problems of traditional generative models and accurately model complex user interactions in a denoising manner. In the forward process, diffusion models progressively corrupt the initial image x_0 with Gaussian noise; in the backward process, they recover the initial image x_0 from the final image x_T. Because the noise added during the forward process follows a Gaussian distribution rather than being randomly generated, based on Markov chain theory, the state of the noise at any moment during the noise addition process can be calculated, making the training of diffusion models controllable; while the backward process is the removal of noise added during the forward process, which is computationally tractable. We propose the AMLDM model based on the diffusion model in the field of image processing.AMLDM simultaneously address the challenges in generative models: the training instability in GANs and the difficulty in computing the posterior distribution in VAEs.

2. Method

As shown in Figure 1, The AMLDM model can be divided into two parts:Forward process: Gaussian noise is gradually added to disrupt the user's interaction history. This process is part of a Markov chain.Reverse process: Noise removal is gradually learned to restore the original data. Similarly, this is also a Markov chain, where the transition probability $p\theta(x_{t-1}|x_t)$ is based on the data distribution generating x_{t-1} given x_t. In the Reverse process, a multi-head attention mechanism is employed to enhance the model's learning capability.

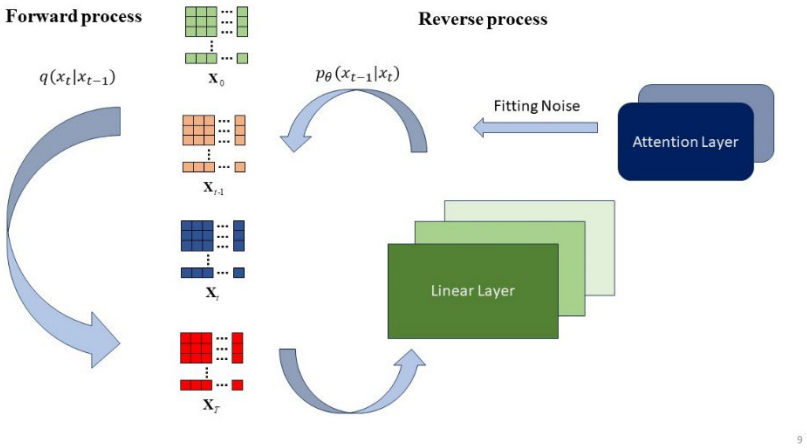

Figure 1: Attention-based Multilayer Linear Diffusion Model (AMLDM)

The Forward process is a continual addition of pre-defined Gaussian noise to the initial data distribution. In other words, over time, the current data distribution is generated by adding pre-defined noise to the distribution of the previous time step. This noise addition method is guided by the theory of Markov chains and possesses "memorylessness" – the calculation of the next state only requires the utilization of the previous state. The Markov chain theory decomposes the complex forward process into a series of discrete states and transition probabilities, thereby simplifying the complexity of the diffusion model. Compared to randomly adding noise, pre-defined noise allows for the precise calculation of the data state at any given time.

To add noise to the state x_{t-1} to transition it to state x_t, the formula is as follows:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \tag{1}$$

Where, $x_t$represents the data distribution at time t, $\beta_t \in (0, 1)$. The mean and variance of the injected noise are determined by β_t, used to control the scale of Gaussian parameter injection. Based on renormalization techniques and the additivity property of two Gaussian distributions, we do not need to iteratively compute every step of the state in the Markov chain, but can directly compute x_t from the initial state x_0:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \tag{2}$$

ISSN 2616-5775 Vol. 7, Issue 4: 56-61, DOI: 10.25236/AJCIS.2024.070408

Where $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \Pi_{i=1}^{t} \alpha_i$, with random noise ϵ, $\epsilon \sim \mathcal{N}(0, I)$.

In the original diffusion model, noise grows linearly with time t. Specifically, the magnitude of noise added in the first step is 0.1, in the second step is 0.2, and in the third step is 0.3. However, for the initial data, after the first step, the noise level is 0.1, after the second step, it becomes 0.3, and after the third step, it reaches 0.6. Since recommendation systems are highly sensitive to noise, excessive noise may lead to imbalanced recommendation outcomes. Therefore, this study adopts a simple linear transformation, making the noise level in the data a linear function of time t.

$$1 - \bar{\alpha}_t = R \cdot \left[N_{max} - \frac{T - t}{T - 1} (N_{max} - N_{min}) \right], t \in 1, \dots, T \tag{3}$$

$R \in (0,1)$ is used to modulate the intensity of noise generation; $N_{max} > N_{min} \in (0,1)$ represent the upper and lower bounds of the noise, respectively. T is the total number of steps for noise generation, while t is the current step. In the formula, the only variable is t, with all other parameters being fixed constants, resulting in noise generation being a linear function of t. During the forward process, the generated noise progressively approaches the set upper limit, N_{max}. Through linear transformation and selecting appropriate noise upper bounds, it ensures smooth noise generation without disrupting the normal operation of the recommendation system.

We choose to use user-item interaction data as the initial data distribution:

$$x_u = \left[x_u^1, x_u^2, \dots, x_u^{|I|} \right] \tag{4}$$

Where $x_u^1 = 1$ represents an interaction between user and item 1.

The Reverse process involves gradually removing the noise added during the forward process, restoring the user's initial interaction history. When the added noise is sufficiently small, we can assume it follows a Gaussian distribution $q(x_{t-1} \mid x_t)$, but cannot progressively fit this distribution because it requires traversing the entire dataset, which would be computationally prohibitive. Similar to the forward process, the backward process is also a Markov chain. We can equivalently substitute $q(x_{t-1} \mid x_t, x_0)$ for the inexpressible distribution $q(x_{t-1} \mid x_t)$.

Expanding $q(x_{t-1} \mid x_t, x_0)$ and substituting into the probability density function of the Gaussian distribution and parameter renormalization, we obtain the mean and variance as x_0:

$$\tilde{\mu}(x_t, x_0, t) = \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}(1 - \alpha_t)}{1 - \bar{\alpha}_t} x_0 \tag{5}$$

$$\widetilde{\beta_t} = \frac{(1 - \alpha_t)(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \tag{6}$$

Since $q(x_{t-1} \mid x_t)$ is unknown, we need to design a neural network to fit $q(x_{t-1} \mid x_t)$, the formula is as follows:

μ_θ and $\Sigma_\theta$represent the mean and variance of the Gaussian distribution obtained through neural network training. Determining the optimization direction by computing the likelihood function of equation (7).

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}\left(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)\right) \tag{7}$$

$$\log p_\theta(x_0) \geq \log p_\theta(x_0) - D_{KL}(q(x_{1:T}|x_0) \mid\mid p_\theta(x_{1:T}|x_0)) \tag{8}$$

We skip the intermediate complex derivation process and directly provide the final results.And the final obtained loss function is (9):

$$Loss = \mathbb{E}_{x_0, \epsilon}(||\epsilon - \epsilon_\theta(x_t, t)||^2) \tag{9}$$

ϵ represents the true generated noise, and ϵ_θ needs to be fitted by a neural network. The smaller the difference between the two, the smaller the loss function, and the more accurate the denoising process. When the model fits A, it is substituted into equation (7) obtain x_{t-1}.

Where $\sigma_t z$ represents the error between simulated noise and actual noise.

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right) + \sigma_t z \tag{10}$$

ISSN 2616-5775 Vol. 7, Issue 4: 56-61, DOI: 10.25236/AJCIS.2024.070408

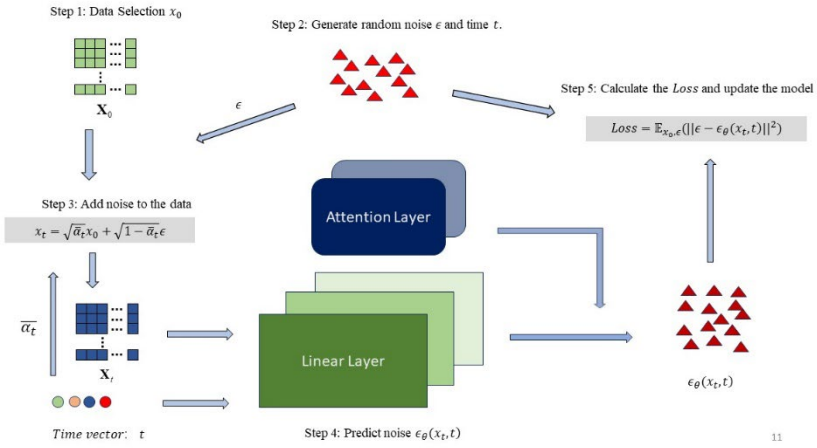

Figure 2: The training process of the AMLDM

As shown in Figure 2, the core component of AMLDM is the multi-attention linear layer, which takes x_t and t as inputs and outputs the fitted noise ϵ_θ.

We designed a multi-attention linear layer, which consists of the commonly used linear layers and attention layers. By utilizing self-attention mechanism, this layer can compute different weights for linear layers at different positions. Combining self-attention and multi-linear layers, we constructed the multi-attention linear layer, whose specific structure is shown in Figure 3.

Assuming there are M linear layer outputs $\{y_1, y_2, ..., y_M\}$, and N attention layers are used, which contain learnable parameters $\{v_1, v_2, ..., v_N\}$, $\{q_1, q_2, ..., q_N\}$, and $\{k_1, k_2, ..., k_N\}$, with input y having a dimension of d: $y=wx$. The calculation of the attention$_i$ layer is as follows:

$$attention_i = softmax\left(\frac{q_i y(k_i y)}{\sqrt{d}}\right)v_i y \tag{11}$$

In the multi-attention linear layer model, q_i (query), k_i (key), and v_i (value) can be the same input (self-attention) or different inputs. In the multi-linear layer model, the output of each linear layer can be used as q, k, and v. In self-attention, q_i and k_i both come from the output of the multi-linear layer, and they are used to learn the dependency relationship of qi on all other k_i, meaning each feature information is a combination of relationships among all other feature information within the group. After calculating all the attention, we take their average and multiply the average result with the output of the linear layer to obtain the final result, which is then output. The final result is the fitted noise. The formula is as follows:

$$attention_i = softmax\left(\frac{q_i y(k_i y)}{\sqrt{d}}\right)v_i y \tag{12}$$

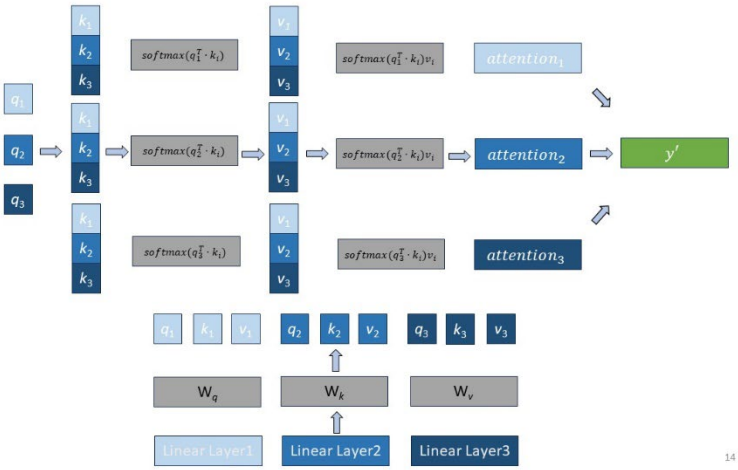

Figure 3: The Structure of the multi-attention linear layer

ISSN 2616-5775 Vol. 7, Issue 4: 56-61, DOI: 10.25236/AJCIS.2024.070408

3. Experiments

In this chapter, to validate the effectiveness of the proposed model, we conducted a series of experiments on multiple publicly available datasets. First, we will introduce the information of the datasets used in the experiments and the methods for evaluating the performance of the algorithms. Then, we will introduce several baseline models for recommendation systems and the experimental parameter settings used for comparison. Finally, we will analyze the experimental results.

3.1. Datasets

To evaluate the performance of the proposed model, we utilized two publicly available datasets. Table 1 summarizes detailed information about the datasets, including the number of users, items, and relationships between users. A brief description of each dataset is provided below:

(1) MovieLens-1m: MovieLens-1m is a classic dataset in traditional recommendation systems, collected and publicly available from the MovieLens website by the GroupLens project. It consists of movie ratings provided by users.

(2) Filmtrust: Filmtrust is a small-scale dataset obtained from the Filmtrust website, known as a classic dataset in social recommendation systems.

Table 1: Descriptions of datasets

	MovieLens-1m	Filmtrust
Ratings	1000209	35497
Users	6040	1508
Items	3952	2071
User Links	None	1853

3.2. Experimental Setup

We compared our proposed model with 5 baseline models, setting parameters according to the original paper, and averaging results over 10 iterations for each experiment.

(1)MultiVAE[1]: This model use a variational autoencoder to capture implicit feedback in collaborative filtering. The core of this model involves incorporating a polynomial likelihood function and using Bayesian inference for parameter estimation.

(2)RecVAE[2]: Building upon MultiVAE, this model adopts a novel composite prior distribution for latent encoding. Regarding the β hyperparameter, this model sets up the β-VAE framework and introduces a training method based on alternate updates.

(3)LightGCN[3]: The most popular GCN (Graph Convolutional Network) models propagate user and item embeddings linearly onto the user-item interaction graph, thus learning embeddings for users and items. In this model, embeddings learned at each layer are weighted and aggregated, serving as the final embeddings. This approach achieves collaborative filtering through neighbor

(4)DGCF[4]: This graph neural network model, which focuses on user intent, models an intent distribution for each user-item interaction and iteratively optimizes to decouple the user intent within the intent distribution.

(5)BPR[5]: Recommendation models commonly used for personalized ranking tasks employ the core idea of Bayesian analysis, optimizing personalized ranking through maximum a posteriori estimation.

For each experiment, I randomly shuffle the dataset and split it into train, validation, and test sets in a ratio of 8:1:1.

3.3. Experimental Results

As shown in Tables 2 and 3, our model exhibits significant performance advantages in terms of recommendation performance compared to five baseline recommendation models on the two datasets.

ISSN 2616-5775 Vol. 7, Issue 4: 56-61, DOI: 10.25236/AJCIS.2024.070408

Table 2: Recommendation performance on MovieLens-1m

Methods	Recall@10	MRR@10	NDCG@10	Hit@10	Precision@10
RecVAE	0.1846	0.4744	0.2795	0.777	0.2142
DGCF	0.1709	0.4536	0.2639	0.7525	0.2058
LightGCN	0.1620	0.4488	0.2590	0.7407	0.2029
BPR	0.1604	0.4431	0.2564	0.7358	0.2026
MultiVAE	0.1718	0.4448	0.2567	0.7629	0.1995
AMLDM	**0.2017**	**0.5106**	**0.3051**	**0.803**	**0.2346**

Table 3: Recommendation performance on Filmtrust

Methods	Recall@10	MRR@10	NDCG@10	Hit@10	Precision@10
DGCF	0.6745	0.4722	0.494	0.7514	0.171
RecVAE	0.6774	0.4835	0.5049	0.75	0.1705
BPR	0.6661	0.4656	0.4873	0.7457	0.1698
LightGCN	0.665	0.4663	0.4881	0.7407	0.1696
MultiVAE	0.6596	0.424	0.4593	0.7357	0.1689
AMLDM	**0.6914**	**0.5097**	**0.5255**	**0.76**	**0.1731**

4. Conclusions

In this study, we introduce a novel recommendation model, namely AMLDM, which is an adaptive generative recommendation model based on the diffusion model. To ensure the accuracy of personalized recommendations, we reduce the noise ratio during the forward process. We conducted empirical validation on two different datasets, and the results demonstrate that AMLDM has significant advantages in recommendation performance.

References

[1] Liang D, Krishnan R G, Hoffman M D, et al. Variational autoencoders for collaborative filtering[C]// Proceedings of the 2018 world wide web conference. 2018: 689-698.
[2] Shenbin I, Alekseev A, Tutubalina E, et al. Recvae: A new variational autoencoder for top-n recommendations with implicit feedback[C]//Proceedings of the 13th international conference on web search and data mining. 2020: 528-536.
[3] He X, Deng K, Wang X, et al. Lightgcn: Simplifying and powering graph convolution network for recommendation[C]//Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval. 2020: 639-648.
[4] Wang X, Jin H, Zhang A, et al. Disentangled graph collaborative filtering[C]//Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval. 2020: 1001-1010.
[5] Rendle S, Freudenthaler C, Gantner Z, et al. BPR: Bayesian personalized ranking from implicit feedback [J]. arXiv preprint arXiv:1205.2618, 2012.