Wide and Deep Autoencoders for Recommender Systems

Atharv Rajendra Jagtap

Advisor: **Dr. Kaustuv Nag**

Department of Computer Science and Engineering
Indian Institute of Information Technology Guwahati

This dissertation is submitted for the degree of
Bachelors of Technology

IIIT Guwahati April 2024

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Atharv Rajendra Jagtap
April 2024

Acknowledgements

I acknowledge Dr. Kaustuv Nag's invaluable guidance and support throughout this project. I also extend my gratitude to all the faculty in the Department of Computer Science and Engineering at the Indian Institute of Information Technology Guwahati for their mentorship.

Abstract

This study proposes a recommendation system that incorporates advanced techniques such as deep learning and autoencoders. Specifically, we propose a novel approach that combines wide and deep models with autoencoders to improve recommendation accuracy. Our method introduces new techniques that incorporate autoencoders, enabling the preservation of essential structures in the encoded feature space by reducing the dimensionality of the encoded features. We evaluate our proposed method on the movie lens dataset, demonstrating its effectiveness compared to existing recommendation algorithms. Furthermore, we provide comparisons with state-of-the-art recommendation systems.

Table of Contents

Chapter 1

Introduction

1.1 What are Recommender Systems?

Recommender systems are information filtering tools designed to predict and present items or content likely to interest users. These systems leverage user data, such as past interactions, preferences, feedback, and item characteristics, to generate personalized recommendations. We employ recommender systems in various online platforms, including e-commerce websites, streaming services, and social media platforms.

1.2 Why Recommender Systems are Essential

In today's digital age, the sheer volume of available content and products has made it increasingly challenging for users to discover relevant items amidst the vast sea of choices. Recommender systems address this challenge by providing personalized recommendations, enhancing user experience, increasing engagement, and driving business revenue. Recommender systems improve user satisfaction and loyalty by alleviating information overload and helping users find items that align with their preferences and interests.

1.3 Objectives of the Thesis

The primary objective of this thesis is to investigate the effectiveness of different recommender system algorithms across various real-world scenarios. Specifically, we aim to:

1. Explore the underlying principles of various recommendation techniques, including collaborative filtering, content-based filtering, and hybrid methods.
2. Evaluate the performance of recommender systems in terms of accuracy, diversity, scalability, and user satisfaction.
3. Investigate the impact of different factors, such as data sparsity, user cold start, and algorithm scalability, on recommender system performance.
4. Propose novel approaches and optimizations to enhance the effectiveness and efficiency of recommender systems.

By achieving these objectives, we aim to contribute to advancing recommender system research and practice, ultimately improving the quality and relevance of recommendations provided to users in diverse online domains.

Chapter 2

Literature Review

2.1 Unsupervised Collaborative Filtering Techniques

2.1.1 User and Item-based Similarity

The user-user and item-item similarity models calculate a similarity score between users or items based on their rating patterns. The cosine similarity between two users u and u' is calculated as:

$$S_{u,u'} = \frac{\sum_j r_{u,j} r_{u',j}}{\sqrt{\sum_j r_{u,j}^2} \sqrt{\sum_j r_{u',j}^2}} \tag{2.1}$$

where $r_{u,j}$ denotes the rating by user u for item j.

The rating prediction for user u on item j is done as:

$$\hat{r}_{u,j} = \sum_{u' \in N(u), j \in R(u')} w_{u,u'} r_{u,u'} \tag{2.2}$$

where $w_{u,u'}$ is the normalized similarity weight between users u and u', and $r_{u,u'}$ is the rating by user u' for jth item. Item-item similarity is calculated similarly by computing cosine similarity between items.

2.1.2 Matrix Factorization [4]

Matrix factorization techniques map users and items to a joint latent factor space, where each user u and item i are associated with a vector p_u and q_i respectively. The

r_{ui} rating is approximated by the dot product $q_i^T p_u$. The optimization objective is:

$$\min_{q_*, p_*} \sum_{(u,i) \in k} (r_{ui} - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2) \tag{2.3}$$

where k is the set of user-item pairs in the training data and λ is the regularization parameter.

2.1.3 Probabilistic Matrix Factorization [5]

Probabilistic matrix factorization scales linearly with the number of observations and maximizes the log-posterior over user and item feature matrices:

$$\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{M} I_{ij}(R_{ij} - U_i^T V_j)^2 + \frac{\lambda_u}{2} \sum_{i=1}^{N} ||U_i||_{frob}^2 + \frac{\lambda_v}{2} \sum_{j=1}^{M} ||V_j||_{frob}^2 \tag{2.4}$$

where U and V are the user and item latent feature matrices and λ_u, λ_v are regularization parameters.

2.1.4 Blind Compressed Sensing [2]

This approach assumes that the item latent factor matrix is sparse, unlike the dense user latent factor matrix. The objective function is:

$$\min_{(U,V)} ||Y - A(UV)|| + \lambda_u ||U||_{frob} + \lambda_v ||V||_{frob} \tag{2.5}$$

where A is the binary mask matrix, Y is the rating matrix, and λ_u, λ_v are regularization parameters.

2.1.5 Matrix Completion [3]

Matrix completion aims to fill the missing entries of a partially observed matrix by solving the nuclear-norm regularized problem:

$$\min ||B - R||_{frob}^2 + \lambda ||R||_{NN} \tag{2.6}$$

where $B = R_{k-1} + M^T(Y - M \cdot R_{k-1})$, M is the binary mask, R is the imputed rating matrix, and Y is the original rating matrix.

2.2 Supervised Collaborative Filtering: Supervised Matrix Factorization

Supervised matrix factorization incorporates user and item metadata to enhance prediction accuracy. Class information matrices W and Q are formed based on user and item metadata. The objective function is:

$$\min_{(U,V,A,C)} ||Y - M(UV)||_{frob} + \lambda_u ||U||_{frob} + \lambda_v ||V||_{frob} + \mu_u ||W - UC||_{frob} + \mu_v ||Q - AV||_{frob} \tag{2.7}$$

where C is the linear map from latent factor space to classification domain, and λ_u, λ_v, μ_u, μ_v are regularization parameters.

Chapter 3

Proposed Method [1]

3.1 Wide Component

The wide component is a generalized linear model:

$$y = \mathbf{w}^T \mathbf{x} + b \tag{3.1}$$

where y is the prediction, $\mathbf{x} = [x_1, x_2, \ldots, x_d]$ is the vector of d input features, $\mathbf{w} = [w_1, w_2, \ldots, w_d]$ are the model parameters, and b is the bias term. It uses cross-product feature transformations defined as:

$$\phi_k(\mathbf{x}) = \prod_{i=1}^{d} x_i^{c_{ki}} \tag{3.2}$$

where $c_{ki} \in \{0, 1\}$ indicates if the i-th feature is part of the k-th transformation ϕ_k.

3.2 Deep Component

The deep component is a feed-forward neural network. One-hot encoding is replaced with an embedding vector $\mathbf{v}_i \in \mathbb{R}^k$ for categorical features. Each hidden layer performs as follows:

$$\mathbf{a}^{(l+1)} = f(\mathbf{W}^{(l)} \mathbf{a}^{(l)} + \mathbf{b}^{(l)}) \tag{3.3}$$

where l is the layer index, $\mathbf{a}^{(l)}$ is the activation vector, $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are weights and biases, and f is the activation function, for instance ReLU.

3.3 Wide and Deep Model

The wide and deep components are combined using a weighted sum:

$$P(Y = 1|\mathbf{x}) = \sigma(\mathbf{w}_{\text{wide}}^T[\mathbf{x}, \phi(\mathbf{x})] + \mathbf{w}_{\text{deep}}^T \mathbf{a}^{(l_f)} + b) \tag{3.4}$$

where σ is the sigmoid, $\mathbf{w}_{\text{wide}}$ and $\mathbf{w}_{\text{deep}}$ are weights, $\phi(\mathbf{x})$ are cross-product transformations, $\mathbf{a}^{(l_f)}$ are activations from the final deep layer, and b is the bias.

The loss function is computed from this joint prediction, and gradients are backpropagated to simultaneously update both wide and deep components.

3.4 Improvements Made Using Autoencoders

3.4.1 Input Processing

To address the challenge of handling sparse input features in the deep component of our model, we employed an autoencoder-based approach. Specifically, we utilized autoencoders to learn dense representations of the sparse input features, thereby reducing the dimensionality of the input space and capturing essential patterns in the data.

The process begins by encoding the sparse input features into dense representations using an autoencoder. Let $\mathbf{x}$ denote the original sparse input features, and $\mathbf{z}$ represent the learned dense representation obtained from the autoencoder. Mathematically, this encoding process can be represented as:

$$\mathbf{z} = f(\mathbf{W}_e \mathbf{x} + \mathbf{b}_e) \tag{3.5}$$

where $\mathbf{W}_e$ and $\mathbf{b}_e$ are the weights and biases of the encoding layer, respectively, and f is the activation function.

3.4.2 For the Wide Model

The encoded dense representation $\mathbf{z}$ is then passed into the deep component of our model. However, it's important to note that this encoding step only applies to the categorical features, typically one-hot encoded. The numerical features remain unchanged.

For the Deep Model

For the wide component of our model, we use the original sparse input features without encoding. This decision preserves the wide component's interpretability and simplicity, which relies on memorizing feature interactions.

3.4.3 Final Integration

In summary, while the deep component benefits from the dense representations learned by the autoencoder, the wide component retains the simplicity and interpretability of the original sparse features. This hybrid approach allows us to leverage the strengths of both components and achieve improved performance in predicting movie ratings.

Chapter 4

Experiments and Results

This Chapter presents the experimental setup and the results obtained from running our models. We conducted a series of experiments to evaluate the performance of the Wide and Deep model both with and without autoencoders. We repeat each experiment ten times and report the average results to ensure the robustness of the results.

4.1 Experimental Setup

We used the MovieLens-100k dataset, which consists of user ratings for movies, as our dataset. The dataset was split into training and testing sets using an $80 : 20$ split ratio. For each experiment, we trained the models on the training set and evaluated their performance on the testing set.

The Wide and Deep model architecture comprised a wide component, which utilized linear models to capture feature interactions, and a deep component, consisting of multiple layers of neural networks. We used TensorFlow to implement the model and trained it using stochastic gradient descent with a learning rate of 0.001.

We report the results in Table 4.1. Overall, the experimental results validate the effectiveness of incorporating autoencoders into the Wide & Deep model for recommendation tasks, leading to improved accuracy and relevance of recommendations.

Table 4.1 Results With and Without Autoencoders

Run	Without Autoencoders	With Autoencoders
1	0.66	0.75
2	0.75	0.78
3	0.65	0.69
4	0.72	0.75
5	0.76	0.70
6	0.75	0.77
7	0.68	0.69
8	0.65	0.71
9	0.72	0.75
10	0.75	0.75
Average	0.71	0.73

Chapter 5

Conclusions

5.1 Integration of Autoencoders

To address the challenge of handling sparse input features in the deep component of our model, we integrated autoencoders. Autoencoders were utilized to learn dense representations of the sparse input features, reducing the dimensionality of the input space and capturing important patterns in the data.

The process involved encoding the sparse input features into dense representations using an autoencoder. This encoding step was applied only to categorical features, while numerical features remained unchanged. The encoded dense representations were then passed into the deep component of our model.

Meanwhile, for the wide component of our model, we retained the original sparse input features without encoding. This decision preserved the interpretability and simplicity of the wide component, which relies on memorization of feature interactions.

In summary, the integration of autoencoders allowed the deep component to benefit from dense representations while maintaining the simplicity and interpretability of the wide component. This hybrid approach leveraged the strengths of both components, resulting in improved performance in predicting movie ratings.

References

[1] Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., et al. (2016). Wide & deep learning for recommender systems. *IEEE Computer Society*. v, 6

[2] Gogna, A. and Majumdar, A. (2015). Blind compressive sensing framework for collaborative filtering. *arXiv preprint arXiv:1505.01621*. v, 4

[3] Hastie, T., Mazumder, R., Lee, J., and Zadeh, R. (2015). Matrix completion and low-rank svd via fast alternating least squares. *arXiv preprint arXiv:1410.2596*. v, 4

[4] Koren, Y. (2009). Matrix factorization techniques for recommender systems. *IEEE Computer Society*. v, 3

[5] Mnih, A. and Salakhutdinov, R. (2007). Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, pages 1257–1264. v, 4