Aviv Ples 318357233
Elad Prager 200865780

Recommendation Systems - Final Project Report

Introduction

The aim of this project was to develop and compare three different models for a recommendation system. The first model served as a baseline, which was a simple WIDE model based on MLP with one dense layer. The second model was based on the "Deep Neural Networks for YouTube Recommendations" paper, with three dense layers, and was considered a DEEP model. The third model was an improvement on the previous two, combining both a wide and deep component, and was based on the "Wide & Deep Learning for Recommender Systems" paper.

The architecture of all models was based on the "Deep Neural Networks for YouTube Recommendations" paper, with both a candidate generation network and a ranking network. The models were trained and tested on a dataset, and the results indicated that the WIDE and DEEP model outperformed the DEEP model, which in turn outperformed the WIDE model.

That combination captures both simple feature interactions that might be missed by a deep model, as well as complex non-linear relationships between the features. Overall, the results showed that jointly training wide linear models and deep neural networks provides a balance between memorization and generalization, leading to superior performance compared to a regular deep network. This highlights the importance of considering both the wide and deep components in the development of such systems in the field of Recommendation Systems.

Anchor Paper

The anchor paper is: **Deep neural network for youtube video recommendation (2016)**

Main objective: The main objective of the paper is to describe the deep neural network architecture used by YouTube for recommending videos to users. The authors are trying to solve the problem of recommending personalized content to each user that maximizes their engagement and watch time on the platform. They achieve this by designing a deep collaborative filtering model that can assimilate many signals and model their interactions with layers of depth. The model is split into two distinct problems: candidate generation and ranking. The authors demonstrate that their deep learning approach outperforms previous linear and tree-based methods for watch time prediction and recommendation systems, in particular, benefit from specialized features describing past user behavior with items.

Architecture: The paper presents a deep neural network architecture for recommendation systems on YouTube that uses both collaborative filtering and content-based features. The model is split into two parts: a candidate generation model that generates a large pool of candidate videos for a user, and a ranking model that ranks the candidates for a personalized recommendation. The architecture uses deep neural networks with multiple

hidden layers to model complex user-item interactions, and also incorporates a variety of features, including user history, video metadata, and video embeddings.

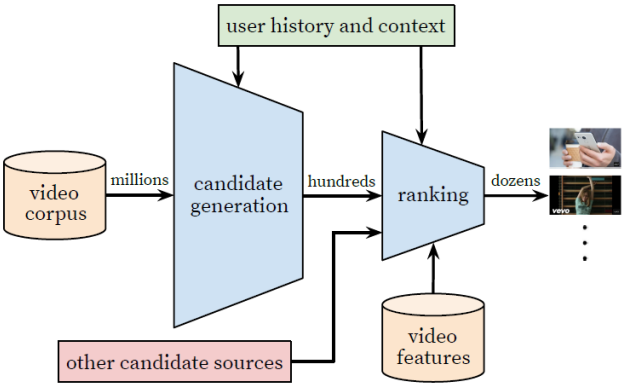

Figure from the anchor paper

Specifically, in our project, we've used a candidate network for generating movie recommendations for a given user, as described in the paper. The candidate network took in user features, such as user information and their watch history, and used a pyramid-shaped structure of three dense layers to process the data. The network was trying to predict the next movie the user will be watching. Thus, the activation function for the output layer during training was Softmax over all movies, and we were using cross entropy as our loss function, while during serving, our network runs a KNN of the N nearest neighbors over the dot product of the user features represented by the last RELU layer, with the best candidate movie features according the highest probability by the Softmax operation.

For ranking the movies, we used a ranking network that took in user features along with the features of one of the candidate movies for that user. This network also used a pyramid-shaped structure of three dense layers. The network was trying to predict the normalized rating a user will give to a given movie (out of the candidates which were outputs of the candidate network). Thus, during training, the output was a Sigmoid representing the predicted normalized ranking a specific user will give to that specific movie, and we were using MSE as our loss function over the predicted rating and the true normalized rating. During serving, the output performs an exponential operation over the output of the final dense layer before the Sigmoid. By running the ranking network over the combination of a specific user with all their N candidate movies (which were outputted from the candidate network), we were able to deliver a list of ranked movies per user.

Improvement

The improvement is based on the paper: **Wide & Deep Learning for Recommender Systems**

Proposed revision architecture: Wide and Deep Learning (WDL) is a type of neural network architecture that combines the strengths of linear models and deep neural networks to improve the performance of recommender systems.

In a WDL architecture, the network is split into two parts: a "wide" part and a "deep" part. The wide part of the network is a linear model that learns feature interactions between different input features. This means that the network can learn to make recommendations based on simple rules. The deep part of the network is a neural network that can learn complex non-linear relationships between features, allowing the network to make more nuanced recommendations.

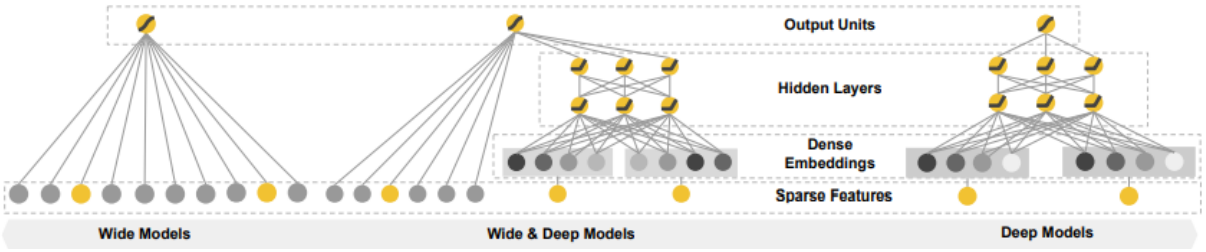

Figure from the improvement paper

Compared to a regular MLP model, a WDL architecture has several advantages for recommender systems. First, the wide part of the network can capture simple feature interactions that may be missed by a deep model. Second, the deep part of the network can learn complex non-linear relationships between features that may be difficult to capture with a linear model. Finally, the combination of both parts of the network can lead to better generalization performance and higher accuracy on new data.

Baseline

We've used a regular MLP model with one dense layer as a baseline for comparison with more complex models. Our regular MLP model is a simple and widely used neural network architecture that consists of input and output layers with one hidden layer of densely connected nodes. We chose to use a regular MLP model as a baseline because it is easy to implement, provides a good starting point for model development, and can help us to establish a baseline level of performance for more complex models. By comparing the performance of more complex models with that of the regular MLP model, we can gain a better understanding of the relative advantages and limitations of different neural network architectures for the task at hand.

Evaluating

Dataset: we've used the Movielens 100K dataset as our main data source. In this dataset, we collected observations for each user by splitting their watch history into multiple windows with a size of 20. The first 19 movies in each window were used to compose the user's history, while the last movie in each window was used as the label for that window. We split the data into a training set (80%) and a test set (20%), and during the training phase, we also used a validation set (20% of the training set). Also, it is worth noting that we've compared the three algorithms (baseline, paper and improvement) on the same train and test set.

The choice of the Movielens dataset was based on the fact that it is a well-known dataset widely used for recommender system research. It contains a large number of user-movie interactions and ratings, which makes it suitable for training and evaluating recommendation models. The dataset is also similar to the "Deep Neural Networks for YouTube Recommendations" dataset in several ways. Both datasets are related to videos, contain information about users and videos, and can be used to construct observations that capture users' preferences. In the Movielens dataset, the rating score can be considered as a measurement of how much a user likes a particular movie, while in the YouTube paper dataset, the watching time was used as a measurement. Additionally, by splitting the user's watch history into windows, we were able to construct similar observations as in the YouTube paper.

Hyperparameter optimization: During the training of both the candidate and the ranking networks, we performed hyperparameter tuning to find the best combination of hyperparameters. This process was conducted for all model types, which are WIDE, DEEP, and WIDE & DEEP. We experimented with different learning rates, such as 0.001 and 0.01, different dense unit sizes including 128->256->512 and 256->512->1024, and different feature embedding dimensions such as 256 and 512. After conducting some experiments, we found that changing the feature embedding dimensions had no significant impact on the performance of the models. Therefore, due to limitations in computing power during the tuning process, we decided to use a fixed feature embedding dimension of 512 for all experiments. Of course, this process can be expanded to check more values or tune over more variables like number of epochs and batch size. We focused on the top three experiments and plotted the loss and accuracy for each epoch to show the model convergence. Finally, we chose the best experiment for evaluation.

Performance metrics: For the candidate network, which outputs a list of candidates, we used three metrics: Hit Rate (HR), Mean Reciprocal Rank (MRR), and Normalized Discounted Cumulative Gain (NDCG). These metrics were evaluated at the top 5 and top 10 candidates. The HR metric measures the proportion of test users for whom at least one of the true positive items was recommended in the top K recommendations. MRR measures the average rank of the first true positive item among the top K recommended items. NDCG is a measure of ranking quality that takes into account the position of each relevant item in the ranking. For the ranking network, which outputs a scalar, we used the Root Mean Squared Error (RMSE) metric. RMSE measures the average deviation of the predicted rating from the true rating.

We chose these metrics because they are commonly used in the literature for evaluating recommender systems. HR, MRR, and NDCG are measures of recommendation quality that take into account the order of recommended items. On the other hand, RMSE is a measure of prediction accuracy that does not consider the order of recommended items. By using a combination of these metrics, we were able to evaluate both the ranking and recommendation quality of our models.

Conclusions

Results: From the results of our experiments, we found that the wide and deep model outperformed the other models in terms of all the evaluation metrics. More specifically, the results indicated that the WIDE and DEEP model outperformed the DEEP model, which in turn outperformed the WIDE model. This indicates that the combination of both wide and deep models is effective in capturing both simple and complex feature interactions, leading to improved recommendation and ranking performance.

Conclusions: We have successfully replicated the architecture proposed in the "Deep Neural Networks for YouTube Recommendations" paper on the Movielens dataset. Our approach followed the original paper as closely as possible, and later we also improved the model by incorporating the "Wide & Deep Learning for Recommender Systems"main technique. Our architecture consisted of two networks, the candidate network that predicts a list of candidate movies for a specific user, and the ranking network that ranks the candidates to produce the final recommendation list. We have experimented with different hyperparameters, including learning rates, dense units sizes, and feature embedding dimensions, for all model types, namely WIDE, DEEP, and WIDE & DEEP. We evaluated the models using various metrics, such as HR, MRR, NDCG, and RMSE, and compared their performances. Our experiments showed that the WIDE & DEEP model outperformed the other models for our recommendation task. Overall, this project demonstrated the effectiveness of deep learning techniques for movie recommendation systems, and highlighted the potential for further improvements through the incorporation of wide and deep learning. Additionally, we have shown that it is possible to reproduce the YouTube recommendation system architecture on the MovieLens dataset and even improve upon it by implementing the idea of wide and deep learning.

Future Work: There are several potential areas for future work to improve our movie recommender system. First, we could extract features from other sources about the movies, such as actors, directors, movie length, etc., and use them as additional features to improve the accuracy of our model. Second, we could experiment with more robust hyperparameter tuning by trying more values and tuning over more variables. This could potentially lead to better performance and more optimal hyperparameters for our model.

Third, implementing an early stopping mechanism, such as the one we used in Assignment 3, could be helpful in preventing overfitting and improving the generalization ability of our model. Fourth, we could experiment with the ranking network output using a discrete classification rating score (1-5) instead of a scalar value. This could potentially improve the interpretability of our model and make the recommendations more intuitive for users.

Finally, we could add cross-product transformation between features, as described in the Wide & Deep paper. This could capture the interactions between the binary features and add nonlinearity to the generalized linear model, potentially leading to improved performance.

APPENDIX

References

Paper “Deep Neural Networks for YouTube Recommendations” Paul Covington, Jay Adams, Emre Sargin (2016)
https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45530.pdf

Paper “Wide & Deep Learning for Recommender Systems” paper
https://arxiv.org/pdf/1606.07792.pdf

Paper “Deep Learning based Recommender System: A Survey and New Perspectives”
https://arxiv.org/pdf/1707.07435.pdf

Paper “The YouTube Video Recommendation System”
https://dl-acm-org.ezprimo1.runi.ac.il/doi/pdf/10.1145/1864708.1864770

Papers With Code “Wide & Deep Learning for Recommender Systems”
https://paperswithcode.com/paper/wide-deep-learning-for-recommender-systems

Github Repository “implementation of personalized movie recommendation using a neural network model”
https://github.com/doggydoggy0101/recommendation_network

Github Repository “Unofficial implementation of recommender systems for YouTube from the paper: Deep Neural Networks for YouTube Recommendations”
https://github.com/hyez/Deep-Youtube-Recommendations

Medium Blog “Wide & Deep Learning for Recommender Systems”
https://medium.com/analytics-vidhya/wide-deep-learning-for-recommender-systems-dc99094fc291

Medium Blog “Youtube’s Recommendation System: Made possible with Deep Neural Networks”
https://medium.com/@dillonmedd1/youtubes-recommendation-system-made-possible-with-deep-neural-networks-61fc2709cd34

TowardsDataScience Blog “Using Deep Neural Networks to make YouTube Recommendations”
https://towardsdatascience.com/using-deep-neural-networks-to