Image and Review based Recommender System

Machine Leaning 2 (DA330) Course Project

Sayan Ghosh[1]

[1]Big Data Analytics 2019-21, Department of Computer Science, Ramakrishna Mission Vivekananda Educational and Research Institute, Belur Math, Howrah, India

Abstract

From online shopping with Amazon or Flipkart to watching movies in Netflix, now-a-days recommender systems play an important role in providing better user-experience and increasing revenue for the web-service providers. Most of the deep learning approaches model the user-item interactions either with product images or with reviews given by users for product recommendation. In this paper, we take into account all three — ratings, item images and reviews given by users, and propose a novel autoencoder-based neural network for recommendation. We also employ spatial attention mechanism to extract features from images and use original ratings and pretrained Albert embedding of the reviews to train the model. We have used Amazon Fashion Dataset (2018) to check our model performance.

1 Introduction

Now-a-days recommender system has become a key component for various service providers to thrive in the market. This technique has widely been adopted in various fields, like e-commerce (Amazon, Flipkart, etc.), online video and song playing apps (Youtube, Spotify etc.) and even in social media (Facebook, Instagram etc.) helping to alleviate information overload and resulting in improved user experience. A glimpse of the growing popularity of recommendation system can be found in a review paper on recommender systems by S. Zhang et al[10].

Traditional recommender systems take into account past user-item interactions in form of explicit feedback like ratings or implicit feedback like click logs, and try to model users' preferences on specific items to recommend an item to a particular user. One of the ancient yet popular method for recommendation is Matrix Factorization[4], which is of this kind. In this method user and item are projected to a common latent space and dot product of the corresponding two latent vectors is computed to get the predicted score and we then try to minimize the squared difference of the observed rating and this predicted score. Several works have been done to increase the effectiveness of matrix factorization, such as integrating it with neighbor-based models[5], combining it with topic models of item content[9] and extending it to factorization machines[7] for a generic modelling of features. Matrix factorization was the default choice until when Xiangnan He et al came up with a neural network architecture for recommender systems, which overcame the problem of depending on a simple computation like inner product in matrix factorization[3]. It deploys a generalized matrix factorization technique and combines it with multi-layer perceptron to model the user-item interactions. Here the authors worked with implicit feedback and modelled the problem as a binary classification problem and trained the network with binary crossentropy loss. After this work several neural architectures for recommender systems came to light. One notable work is AutoRec[8]. This is an autoencoder-based collaborative filtering model, which considers each item(user) as a vector of ratings of dimension equal to the number of users(items) and reconstructs the same. The network is trained using regularized mean squared error, where the loss is computed only with respect to the observed ratings and recommendation is done using the predicted ratings for unobserved users(items). Next came CATA++[1], which is an advanced dual attentive autoencoder-based collaborative filtering model. Here two parallel autoencoders derive latent representations for user and item, and the interaction score is obtained using these two latent representations. In this architecture, the two autoencoders and the interaction score prediction module are alternately trained using binary crossentropy loss function and weighted regularized mean squared error respectively.

In our work, we deal with explicit feedback (ratings given by users) and adopt many of the ideas mentioned above to model the user-item interactions. Most importantly we make use of the images of the items and reviews given by customers to make our model robust. In one of the subsequent sections we discuss the model architecture in detail.

2 Description of Data

We have utilized Amazon Fashion Dataset (2018) * for our work. It is an open source dataset which contains information about the outfit (clothes, shoes, belts, bags etc.) sold in Amazon. This dataset has mainly two parts —

1. Transaction File: It contains information about the following attributes: (a) user ID, (b) item ID, (c) user name, (d) helpfulness votes, (e) review, (f) summary of review, (g) rating, (h) review date, (i) unix timestamp of review

2. Item metadata file: It contains information about the following attributes: (a) item ID, (b) title, (c) features, (d) description, (e) price, (f) image link,

*Source of Dataset: https://nijianmo.github.io/amazon/index.html

(g) users who bought, (h) users who viewed, (i) sales-rank, (j) brand, (k) categories, etc.

We preprocessed the data and retained only 4 attributes in the transaction file, namely user ID, item ID, rating and review. We split this data into training and validation dataset such that all the users in the validation dataset are present in the training data as well. From the image links in the metadata we downloaded the images of the items and sub-directorized the images according to item ID. Finally the ratings (which could originally be any one of the 5 integer values 1, 2, 3, 4, 5) are rescaled to a range from -1 to 1 and the images are resized to 40×40×3 before passing into the network.

3 Methodology

In this section we discuss the formulation, model architecture and loss functions used for training the network.

3.1 Formulation

We formulate our problem as a regression problem, where the task is to predict the rating that a particular user will give to a particular item. The inputs to the model are user ID, item ID and image of the item. At inference time the output of the model is a value between -1 and 1.

3.2 Model Architecture

Our model architecture has the following key features:

1. It has two main components — an autoencoder for reconstructing images and a rating prediction module.

2. The image autoencoder is made up of only convolutional and transpose-convolutional layers, where the structure of the decoder is a mirror image of that of the encoder. The encoder consists of two convolutional layers — first one with 32 kernels each of spatial dimension 4×4, stride 3 and no padding, and the second one with 16 kernels each of spatial dimension 3×3, stride 1 and no padding.

3. For the rating prediction module, we pass in as input the one-hot encoded vectors of user and item. In our case the number of users is 614505 and number of items is 132006. The 614505 and 132006 dimensional vectors are passed through two different embedding layers, the outputs of each of which is a vector of size 16. The outputs of the embedding layers are concatenated and passed through two dense layers having 50 and 30 units respectively. At the end we have a dense layer with one unit followed by tanh activation, which gives a value between -1 and 1.

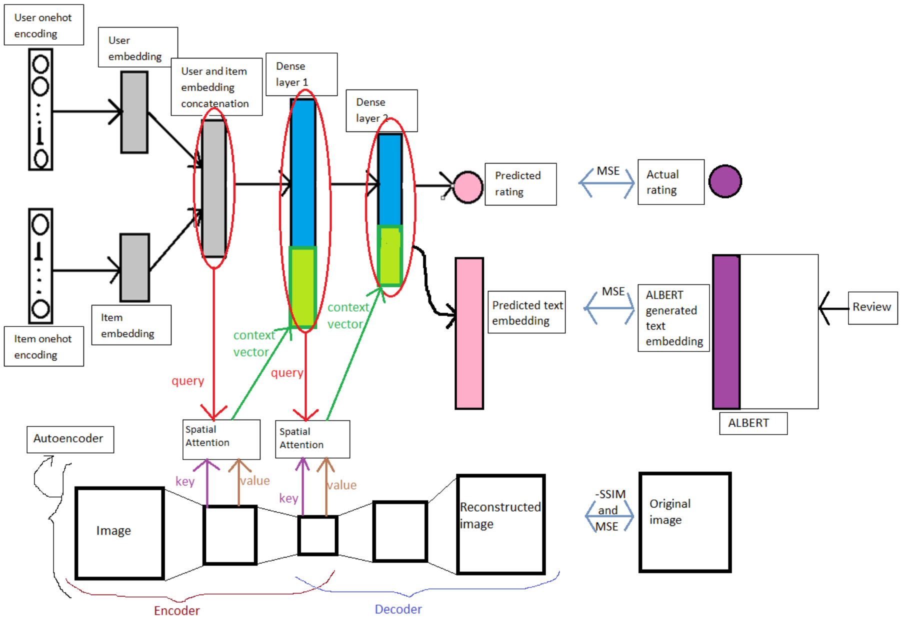

Figure 1: Model Architecture

4. In online shopping the image of a particular item do influence our will to buy it. So we are interested to include that visual information in predicting the rating of an item. For that we feed in information from the encoder layers of the autoencoder to the rating prediction module. We deploy spatial Bahdanau's Attention[2] to extract visual information, where the preceding dense vectors in the rating prediction module works as query and the keys and values are the pixels of the feature maps in the encoder. The output of the attention layer is then concatenated with the next dense layer in the rating prediction module. This method helps to focus on specific visual attributes of the image.

5. Rating and review go hand in hand — if a person gives good review about a product then naturally he/she will give good rating to the product also and in a similar way bad review will imply low rating. So review carries important information about how a person rates a particular item. Now, review and rating are obtained once a person buys a product. So reviews cannot be used for recommendation. But we can use the reviews in the training data to train our model. The way we go about doing this is by introducing a branch with one fully connected layer of 768 units connected to the last intermediate dense layer of the rating prediction module. On the other hand, we use pretrained 'ALBERT[6] for sequence classification' to derive an embedding of the reviews. Minimizing the loss between this

derived embedding and the output of the fully connected layer helps the model to learn better. At test time we can simply discard this branch.

Figure 1 is a schematic of the model.

3.3 Loss functions

In each epoch we train our model in two phases — in the first phase we train the autoencoder and in the second phase we train the rating prediction module along with the encoder and the branch introduced for matching text embedding. So in these two phases we use two different loss functions:

1. In phase I, we use a weighted combination of negative of Structural Similarity (SSIM) index[11] and mean-squared error (MSE) as loss function. Although SSIM works well for reconstruction of images compared to MSE, but we observed that using SSIM alone does not get back exact colour in the reconstructed images as in the original images. MSE tries to compensate that to some extent.

2. In phase II, we use a weighted combination of the mean-squared errors of the predicted and original ratings, and of the predicted and ALBERT generated text embeddings of the reviews.

4 Result

We present the results based on the performance of our model on the validation dataset. The evaluation matrix is mean-squared error (MSE) calculated based on the predicted rating and original rating, the maximum value of which is 4 and minimum value is 0. We trained our model for 30 epochs and our final validation loss is ≈ 0.441042. Figure 2 shows the plot of change of validation loss during training.

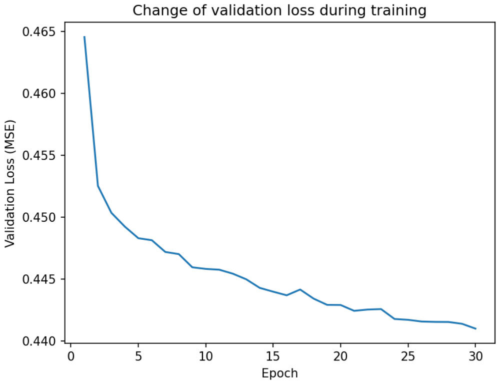

Figure 2: Change of validation loss during training

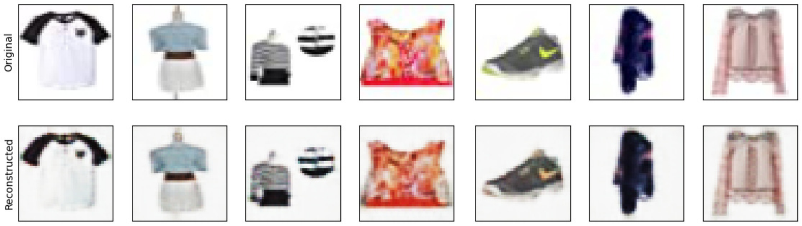

Figure 3: Images reconstructed by trained autoencoder

To demonstrate that our autoencoder has learnt something useful, we present some pictures from the dataset along with their reconstructed version (see Figure 3)

5 Conclusion

The architecture we proposed in this paper is quite generic. People can use this backbone and add further complexities to the model by introducing additional fully connected and convolutional layers, deploying different form of advanced attention mechanism etc. In this project we discussed how we can use the visual and textual information in recommender system. There is another important aspect — time. A person's preference changes with time, which can influence his rating strategy. So there is a scope of investigation about how to incorporate time information in this architecture to get better predictions of ratings.

References

[1] Meshal Alfarhood and Jianlin Cheng. "CATA++: A Collaborative Dual Attentive Autoencoder Method for Recommending Scientific Articles". In: *IEEE Access* 8 (2020), pp. 183633–183648. ISSN: 2169-3536. DOI: 10.1109/access.2020.3029722. URL: http://dx.doi.org/10.1109/ACCESS.2020.3029722.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. arXiv: 1409.0473 [cs.CL].

[3] Xiangnan He et al. "Neural Collaborative Filtering". In: (2017). DOI: 10.1145/3038912.3052569. URL: https://doi.org/10.1145/3038912.3052569.

[4] Y. Koren, R. Bell, and C. Volinsky. "Matrix Factorization Techniques for Recommender Systems". In: *Computer* 42.8 (2009), pp. 30–37. DOI: 10.1109/MC.2009.263.

[5] Yehuda Koren. "Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model". In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '08. Las Vegas, Nevada, USA: Association for Computing Machinery, 2008, pp. 426–434. ISBN: 9781605581934. DOI: `10.1145/1401890.1401944`. URL: `https://doi.org/10.1145/1401890.1401944`.

[6] Zhenzhong Lan et al. *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*. 2020. arXiv: `1909.11942 [cs.CL]`.

[7] Steffen Rendle. "Factorization Machines". In: *Proceedings of the 2010 IEEE International Conference on Data Mining*. ICDM '10. USA: IEEE Computer Society, 2010, pp. 995–1000. ISBN: 9780769542560. DOI: `10.1109/ICDM.2010.127`. URL: `https://doi.org/10.1109/ICDM.2010.127`.

[8] Suvash Sedhain et al. "AutoRec: Autoencoders Meet Collaborative Filtering". In: (2015). DOI: `10.1145/2740908.2742726`. URL: `https://doi.org/10.1145/2740908.2742726`.

[9] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. "Collaborative Deep Learning for Recommender Systems". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '15. Sydney, NSW, Australia: Association for Computing Machinery, 2015, pp. 1235–1244. ISBN: 9781450336642. DOI: `10.1145/2783258.2783273`. URL: `https://doi.org/10.1145/2783258.2783273`.

[10] Shuai Zhang et al. "Deep Learning Based Recommender System". In: *ACM Computing Surveys* 52.1 (Feb. 2019), pp. 1–38. ISSN: 1557-7341. DOI: `10.1145/3285029`. URL: `http://dx.doi.org/10.1145/3285029`.

[11] Wang Zhou et al. "Image Quality Assessment: From Error Visibility to Structural Similarity". In: *IEEE* 13.4 (Apr. 2004).