Computer Recommendation System - CompGAN

authors

Department of Electrical and Computer Engineering, Stony Brook University, NY, USA-11790

Abstract

Generative Adversarial Networks (GANs) are one of the most innovative ideas proposed in this decade. At its core, GANs are an unsupervised model for generating new elements from a set of similar elements. For instance, to produce original face pictures given a collection of face images or create new tunes out of preexisting melodies. GANs have found applications for image, text, and sound generation, being at the core of technologies such as AI music, deep fakes, and content-aware image editing. Besides pure generation, GANs have also been applied to transforming images from one domain to another and as a means for style transfer. To add one more application, they fit as an intelligent data augmentation technique for semi-supervised learning.A GAN, or Generative Adversarial Network, is a generative model that simultaneously trains two models: a generative model that captures the data distribution, and a discriminative model that estimates the probability that a sample came from the training data rather than generative model.

KEYWORDS: Top-N recommendation system, generative adversarial networks

1 Introduction

The promise of deep learning is to discover rich, hierarchical models [2] that represent probability distributions over the kinds of data encountered in artificial intelligence applications, such as natural images, audio waveforms containing speech, and symbols in natural language corpora. So far, the most striking successes in deep learning have involved discriminative models, usually those that map a high-dimensional, rich sensory input to a class label [14, 22]. These striking successes have primarily been based on the backpropagation and dropout algorithms, using piecewise linear units [19, 9, 10] which have a particularly well-behaved gradient . Deep generative models have had less of an impact, due to the difficulty of approximating many intractable probabilistic computations that arise in maximum likelihood estimation and related strategies, and due to difficulty of leveraging the benefits of piecewise linear units in the generative context.

Generative Adversarial Networks (GANs) is a generative model which uses an adversarial minimax game between a generator and a discriminator where the generator focuses on capturing the distribution of real observed data, to generate adversarial samples and fool the discriminator; meanwhile, the discriminator attempts to distinguish whether the inputted sample is from the generator or not. GANs have found applications for image, text, and sound generation, being at the core of technologies such as AI music, deep fakes, and content-aware image editing. Besides pure generation, GANs have also been applied to transforming images from one domain to another and as a means for style transfer. To add one more application, they fit as an intelligent data augmentation technique for semi-supervised learning.

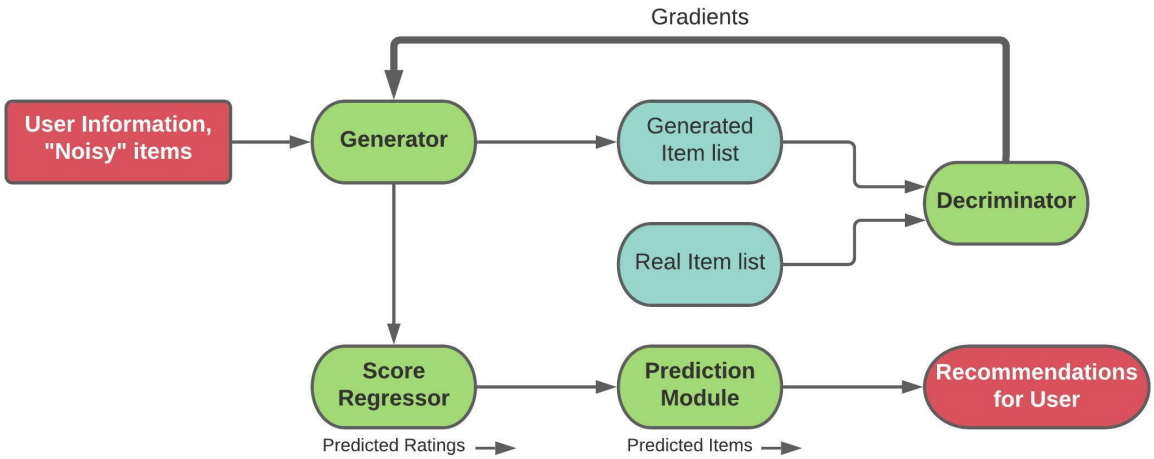

Figure 1: Block diagram of computer architecture.

2 CompGAN architecture

2.1 Why, Where, Problem

2.1.1 Why use a GaN for Recommender systems (RS)?

Data sparsity - trained to generate data by learning the underlying probability distribution.
Generative model - be able to generate "new and better solutions".

2.1.2 Where are GANs successful?

A lot of success in the Computer Vision industry.
Recommendation Systems catching up - IRGANs, CFGANs etc.

2.1.3 Problems faced by recommender system in general:

Data sparsity – fewer user and or item information – insufficient information makes recommendation task difficult.
Data noise – natural (user bias) and malicious noise (intentional).

2.2 Training

Since this RS can be applied to other solutions as well, I have used the term users and items to describe applications and laptops respectively. The user-item interaction is represented using a 0/1 indicator matrix where 0 means that the user u has not seen/used an item i before and 1 means otherwise as we can see in the figure 1. This representation is called implicit feedback matrix.

3 CFGAN

The Generator is trained on the implicit user - item feedback matrix, where m is the number of users and n is the number of items. The Generator's task is to learn to predict the likelihood of an item i to be recommended to a given user u, i.e. it learns the conditional probability of observing an item i given a user u, Pr(I=i—U=u). It takes in a user id u and a noise vector(n items with random values between 0 and 1) and after each step, outputs values between 0 and 1 for each of the n-items. The Discriminator is trained to differentiate between a generated item vector and a real item vector ¡i¿Thus, it is a binary classification problem with a binary cross entropy loss to minimize. It's loss function is given as:

$$J_G = \sum_u (log(1 - D(\hat{r_u} \odot (pos - mask + neg - mask)) + \alpha \sum_j (x_j - \hat{x_j})^2$$

where (x_j, x_j) are the original and ground truth item

4 Generator and Discriminator trained jointly:

- Train on implicit feedback user - item matrix (m, n)
- Minimize Generator loss: Objective function of G and reconstruction loss
- Minimize Discriminator loss: Binary cross entropy
- Backpropagation
- Reconstruction Loss:
  1. MSE of generated items and ground truth
  2. Don't want generator to generate all the items for recommendation – makes no sense.
     (a) Sample some negative items and add to user's list
     (b) Create a negative mask
     (c) Mask the negative items during reconstruction loss calculation

Score Regressor is trained after training the GAN using the items generated by the Generator. If for user u ground truth item list has item set and the generator recommends then we need to impute the ratings for items.

- Train on user information + item information
- Minimize the MSE
- Backpropagation

4.1 Training Steps

Scrape the data from the website - created scraping software (scraper.py) which takes a list of urls as input and outputs a csv file. Rules for handling missing delimiiter and missing cloumns have been added after analysis. Now the data to be process and encoded using ordinal encoding.

1. Encoded (Ordinal encoding):
2. application name is user_id
3. laptop name is item_id
4. cpu is cpu_id
5. ram is ram_id
6. gpu is gpu_id
7. storage is storage_id
8. battery is battery_id

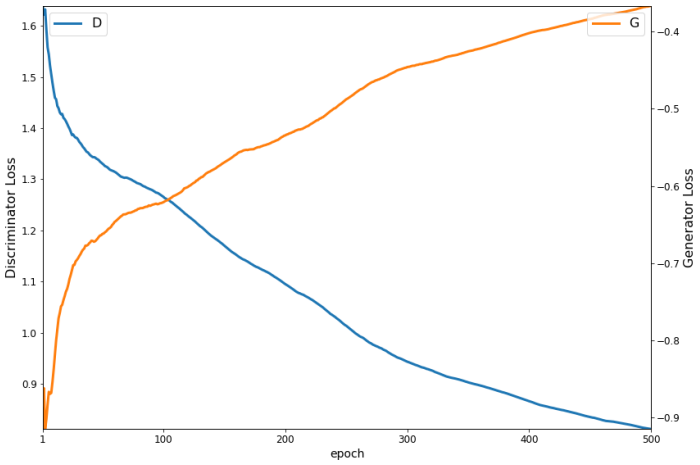

Figure 2: Generator and Discriminator Losses.

The normalized 'score' column using min-max scaling for each categorical value of the application field as scores have different scales per category for example: Futuremark 3DMark 11 has scores like thousands whereas Bioshock Infinite has values in hundreds. Finally the scaled score field are on a 0-5 scale. Data fetching functions for the GAN input is an implicit matrix of size (n_u, n_i) where n_u, n_i are the number of users and items respectively. For the Precision module (ScoreReg) is (n, d) represent each row is (user_id, item_id, item_features) and n is the number of all different user_id, item_id pairs.

Loss function converges for both the Generator and Discriminator but we use the generator to predict an item list given a user_id. Evaluation with implicit matrix that Generator generates the unordered item(laptop) list per user (application). The Precion, Recall, NDCG metrics used to evaluate recommendations on validation set. After that it was trained a prediction module like ScoreReg which predicts a rating given a user_id, item_id and item_features. The Implementation for user_id 'u', A was the original item_list and B is what the Generator generates, then we don't yet have ratings for items in B. Used mean shift for these items. The rest items A and B are 0 in the user-item explicit matrix (similar to implicit but instead of 1's have ratings). After training ScoreReg, we get predictions for user-item pairs in validation set and then sort items based on predicted ratings. Evaluate the predictions original (user item_list) representation with the user and predicted item_list.

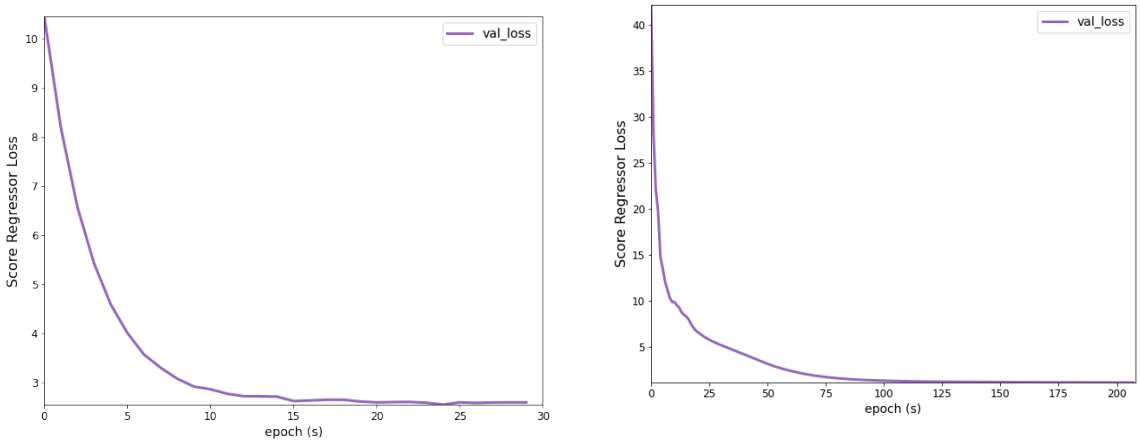

Figure 3: Two different examples for the scorereg loss

5 Results

Recall is low for laptop and movie sets and a predicted item is 'relevant' if it is in the user's ground truth item list. The relevant items / len(set of relevant (ground truth) items). This implies low recall means, the recommendations are new items. Low recall observed in other papers as well.

6 References

1. Hinton, G., Deng, L., Dahl, G. E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., and Kingsbury, B. (2012a). Deep neural networks for acoustic modeling in speech recognition. IEEE Signal Processing Magazine, 29(6), 82–97.

2. Hinton, G. E., Dayan, P., Frey, B. J., and Neal, R. M. (1995). The wake-sleep algorithm for unsupervised neural networks. Science, 268, 1558–1161.

3. Hinton, G. E., Osindero, S., and Teh, Y. (2006). A fast learning algorithm for deep belief nets. Neural Computation, 18, 1527–1554.

4. Hyv¨arinen, A. (2005). Estimation of non-normalized statistical models using score matching. J. Machine Learning Res., 6.

5. Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y. (2009). What is the best multi-stage architecture for object recognition? In Proc. International Conference on Computer Vision (ICCV'09), pages 2146–2153. IEEE.

6. Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y. (2009). What is the best multi-stage architecture for object recognition? In Proc. International Conference on Computer Vision (ICCV'09), pages 2146–2153. IEEE.

7. Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In Proceedings of the International Conference on Learning Representations (ICLR).

8. Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto.

9. Bengio, Y., Mesnil, G., Dauphin, Y., and Rifai, S. (2013a). Better mixing via deep representations. In ICML'13.

10. Bengio, Y., Yao, L., Alain, G., and Vincent, P. (2013b). Generalized denoising auto-encoders as generative models. In NIPS26. Nips Foundation.

11. Bengio, Y., Thibodeau-Laufer, E., and Yosinski, J. (2014a). Deep generative stochastic networks trainable by backprop. In ICML'14.

12. Michael P. O'Mahony, Neil J. Hurley, and Guénolé C.M. Silvestre. 2006. Detecting noise in recommender system databases. In Proceedings of the 11th international conference on Intelligent user interfaces (IUI '06). Association for Computing Machinery, New York, NY, USA, 109–115.

13. Dong-Kyu Chae, Jin-Soo Kang, Sang-Wook Kim, and Jung-Tae Lee. 2018. CFGAN: A Generic Collaborative Filtering Framework based on Generative Adversarial Networks. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM '18). Association for Computing Machinery, New York, NY, USA, 137–146.

14. Gao, Min, Junwei Zhang, Junliang Yu, Jundong Li, Junhao Wen and Qingyu Xiong. "Recommender Systems Based on Generative Adversarial Networks: A Problem-Driven Perspective." ArXiv abs/2003.02474 (2021): n. pag.

15. He, Xiangnan, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua. "Neural Collaborative Filtering." Proceedings of the 26th International Conference on World Wide Web (2017): n. pag.

16. Wide Deep Learning [https://ai.googleblog.com/2016/06/wide-deep-learning-better-together-with.html]

17. Zhang, Shuai, Lina Yao, Aixin Sun and Yi Tay. "Deep Learning Based Recommender System." ACM Computing Surveys (CSUR) 52 (2019): 1 - 38.