Recommendation Systems Approaches on the Netflix Prize Data Set

Cody Blakeney
Email: cjb92@txstate.edu

Samuel Teich
Email: st1289@txstate.edu

***Abstract*—Recommendation systems play an increasingly vital role in the ability to make informed decisions in modern societies, given the ever growing deluge of data which might inform those decisions. Without the ability to quickly choose from among a few well selected options ordinary activities from applying to jobs to searching the internet, buying household goods, or reading the latest news might become difficult, deceptive, or even dangerous. Given the many relatively recent major advances in machine learning techniques, especially in relation to deep learning, it is important to understand how recommendation systems have changed and improved. In this paper we investigate approaches to recommendation systems using deep learning.**

1. Introduction

Data characterizes the modern world. Corresponding to exponential growth in computing power over time, and the related proliferation of sensors and interactions with computing systems, the volume of recorded data has exploded. It is estimated that in 2010 there was more than 1 Zettabyte of recorded unstructured data, and that as of 2018 there exists more than 2 Zettabytes structured data and 20 Zettabytes structured data [2][7]. While much of this deluge is comprised of highly specific and narrowly specialized data, not useful for making decisions at a personal level, there still is a wealth of data - far more than is human interpretable - which could be used to inform such decisions. Recommendation systems are predictive tools designed specifically for this use case, helping pare from a glut of options a select few choices based on available data.

The Netflix Prize competition [1], started in 2006 and finalized in 2009, is, along with the publishing of the ImageNet dataset in 2009 [3], credited as one of the events that encouraged the perception of a revolution in the field of machine learning both among professional audiences and the public at large [6]. In the near decade since team BellKor's Pragmatic Chaos won the Netflix Prize competition [1], rapid advances, especially in Deep Learning, have continued to push boundaries in machine learning and recomendation systems. The Netflix dataset remains an important benchmark dataset for recomendation systems.

In this paper we will discuss recent improvements in the field of recomendation systems and attempt to benchmark some of those improvments on the Netflix dataset. Section 2 will cover some important backround material on the field of recomendation systems. Section 3 will discuss some recent and related results. Section 4 will talk about our experimental setup, while Section 5 will include the specific experiments carried out. Section 6 discusses our results, conclusions, and will cover our expectations for future work.

2. Background

2.1. Recommendation Systems

Recommendation systems predict the preferences of users in regard to new items or objects, in the form of a ranking of items or objects, rating items or objects, or classifying items or objects [8]. There exist a wide variety of established strategies to learn prediction models for recommendation systems, generally fitting into three categories: collaborative filtering, content based, or hybrid systems. Collaborative filtering based recommendation system models learn based on user - item/object interaction, while content based recommendation system models learn based on intrinsic user and item/object characteristics. Hybrid recommendation system models learn based on a combination of the two approaches.

2.2. Deep Learning

Deep learning is a subset of machine learning wherein models learned contain multiple, often heiarchical, abstractions or representations. While there are many approaches to deep learning, in this paper we employ deep feedforward neural networks - feed forward neural networks with one or more hidden layers. A general interpretation of the hidden layers in a deep feedforward neural network is that each layer learns a slightly more complex representation of the data than the layer before it.

2.3. Embedding Layers

Embedding layers encode information about a set of objects and the relationships between those objects in a latent space - generally with fewer dimensions than the original space representing the objects [5]. For example, given a corpus of text with a large vocabulary V, it may be desireable to represent $v \in V$ such that $\mid rep(v) \mid = k$, where $k \ll \mid V \mid$ and that if $v, x \in V$ occur together more frequently than $v, y \in V$, v will be closer to x than y.

2.3.1. Neural Network Embedding Layers.

2.3.2. SVD Embedding Layers. Singual Value Decomposition is a form of Matrix Factorization popular in tradition recommendation systems. The idea is to start with a matrix A where each entry represents an interaction. In the case of a User-movie embedding (here after refered to as user embedding), the rows of A represent a user and the columns represent a movie. Every entry is either the users rating of that movie, or zero. You then decompose your matrix to the product of three matrices $A = U\Sigma V^t$ A is an $M \times N$ matrix U is a $M \times M$ matrix Σ is an $M \times N$ rectangular diagnal matrix and V is a $N \times N$ unitary matrix. To get a reduced approximation of A we simply decide how many k features we want and keep the k most important features of Σ. Multiplying $U\Sigma$ will result in A of rank k. In tradtional recommendation systems the multipying the reduced rank Σ with all three matrices would result in a new approximation of A with values filled in from the reduced rank approximation. The predicted ratings are then looked up in the matrix for an item. For our use case we were only intrested in the reduced space representatoin of the matrix not transforming it back so we kept only $U\Sigma$

3. Related

test

3.1. Takeaway From Related Work

test

4. Approach

Our approach to investigating the architecture for deep learning recommendation systems, was to first decide on a baseline with which we could compare the effectiveness of our techniques. After finding a benchmark we reviewed the literature we had assembled to decide which design was most applicable to our dataset, and our ability to implement.

We knew from the Netflix competition that the winners had an RMSE of 0.8567. We also wanted a way to compare our engineered solution, to one that simple took all the data Netflix provided and used it as inputs on a naive feed-forward neural net.

The theme of all the papers we read covering the various methodologies for applying deep learning to recommendation systems, was creating more meaningful representations of the users, the items they interacted with, and context around the item. Many of the models we read about involved many complex hierarchical layers. While many of these exotic arrangements made sense, we were not sure we had the technical expertise to implement them in the allotted time. We settled on a technique called Neural Collaborative Filtering [4]. Neural Collaborative Filtering or NCF is a process that involves creating a hybrid of traditional recommendation system and deep learning. As you can see from figure 1 it combines user vectors and item vectors from matrix factorization as well as user and item embedding

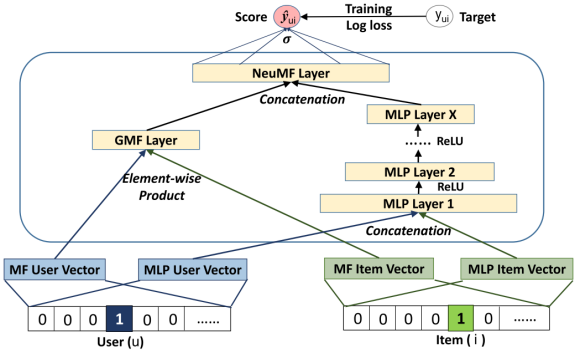

Figure 1. NCF diagram

layers as inputs to the neural network. In order to try to replicate this architecture we would need to need to find appropriate Matrix Factorization representations for our users and our movies, as well as create embedding layers for for them.

5. Setup

5.1. The Netflix Prize Dataset

The Netflix dataset is formatted as a set of 100480507 rating events, each event including the user, movie, and date corresponding to the rating and the 1 - 5 score of the rating itself. In all, the dataset includes ratings from 480189 anonomyzed users in reference to 17770 individual movies and tv-show episodes.

5.2. The IMDB Dataset

The IMDB dataset is refreshed daily and is comprised of seven seperate sub-datasets, of which only three were used:

- title.basics.tsv - Basic information on titles in the IMDB catalog
- title.principals.tsv - Principal cast and crew for each title in the IMDB catalog
- name.basics.tsv - Basic information for cast and crew

In total, 2290948 different titles for movies and tv-show episodes are included in the dataset and 8469380 different names corresponding to cast and crew.

5.3. Experimental Setup

Experiments were conducted using Tensorflow-gpu 1.8.0, Keras 2.1.4, and CUDA 9.0. The machine on which the experiments ran had 32g ram, an i7-6700K CPU, and a GeForce GTX 1080 GPU.

6. Experiments

6.1. Characteristics of the Netflix Dataset

6.2. Naive Feed-forward Deep Neural Networks

What we call our Naive Feed-forward network is the benchmark we made to test the efficacy of the models we were training. We tested several versions before deciding on a final candidate Naive Model. We had a categorical classifier, a regression model, and another categorical model with a better layer design.

6.2.1. Naive Categorical Model. Our first model had inputs of one hots for movies and users, mean, median and variance of the current movie calculated from the training set, and time represented as a month input, a day input, and a year since beginning input. This model had two hidden layers with 128 nodes each using relu activation and a softmax output to guess categorically what the rating was for that user. With this configuration the model converged to about 38% accuracy on our validation set. Because of the distribution of ratings in our data set this is about as good as always guessing the rating 4 for every movie. Not great, but better than randomly chance.

6.2.2. Naive Regression Model. For training the problem as a regression model we learned we had to be very careful with the layer selection. Too few layers and inadequate funneling down of the layers towards the output layer resulted in a model that was hopelessly lost at trying to find a gradient. We finally had success with a model using a hidden layers of size 512, 64, 8, and a linear output layer. After training we were surprised because it had a very good RMSE score. Closer inspection showed that the model was predicting a floating point value that was always between 3.4 and 3.6, very close to the mean value of the data set. When we rounded the value and calculated the accuracy and RMSE it was only 21% accurate with and RMSE of 1.203.

6.3. Neural Network Embeddings on the Netflix Dataset

6.4. Neural Network Embeddings on the IMDB Dataset

6.5. SVD Embeddings on the Netflix Dataset

In [4] they use a process the call GMF or Generalized Matrix Factorization where they use gradient descent to train their embedding to learn the most important latent features

When considering the methods for representing User and Movie matrix factorization embeddings, we decided on using Single Value Decomposition, primarily because it is implemented in the scipy package, and could operate on sparse matrices. This was important because the original matrix we would be decomposing would be of size 440,000 x 17776 and would not possibly fit into memory.

6.6. Deep Feed-forward Neural Networks with Embedding Layers

6.6.1. SVD Deep Embeddings. We needed some way to evaluate how useful the information presented by the SVD Embeddings would be to a deep learning model. We decided to train models on combinations of embeddings and one hots only, without other contextual information like, date of review, actor, genre, etc to see how useful these representations would be. This would tell us, given only a representation of likeness to other users we had computed, how well we could predict their tastes. If we could beat the threshold of randomly guessing or always rating 4, the marks set by our benchmark naive model, than the matrix factorization embeddings would be considered a success.

To test this we first trained to convergence a model that took as inputs both the User SVD embedding and a one hot for its movie id. It was during training this model we experimented with adding more layers and funneling their output as mentioned in [4] with hidden layers of our input size (17826), 1000, 100, 10, and a softmax output layer we were able to achieve an accuracy of around 42%

We also tested using both the User SVD and the Movie SVD as the inputs. This would shrink the input size down from 17826 to 125. Training a model that again had 4 hidden layers we were able to reach an accuracy of 45%. Clearly the representations did have value.

Unfortunately we were not able to also train a model using user one hots and movie embeddings, as the input size made computation prohibitively dificult.

6.7. Hierarchical Architectures

7. Conclusions

7.1. Experimental Discussion

7.2. Future Work

References

[1] [Online]. Available: https://www.netflixprize.com/index.html

[2] J. R. David Reinsel, John Gantz, "Data age 2025: The evolution of data to life-critical," 04 2017. [Online]. Available: https://www.seagate.com/files/www-content/our-story/trends/files/Seagate-WP-DataAge2025-March-2017.pdf

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, "Imagenet: a large-scale hierarchical image database," pp. 248–255, 06 2009.

[4] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural Collaborative Filtering," 2017. [Online]. Available: http://arxiv.org/abs/1708.05031

[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[6] H. Quentin, "Reasons to believe the a.i. boom is real," 03 2018. [Online]. Available: https://www.nytimes.com/2016/07/19/technology/reasons-to-believe-the-ai-boom-is-real.html

[7] L. Rizzatti, “Digital data storage is undergoing mind-boggling growth,” 09 2016. [Online]. Available: https://www.eetimes.com/author.asp?section_id=36&doc_id=1330462

[8] S. Zhang, L. Yao, and A. Sun, “Deep learning based recommender system: A survey and new perspectives,” *CoRR*, vol. abs/1707.07435, 2017. [Online]. Available: http://arxiv.org/abs/1707.07435