Skolkovo Tech

Skolkovo Institute of Science and Technology

SKOLKOVO INSTITUTE OF SCIENCE AND TECHNOLOGY

INTRODUCTION TO RECOMMENDER SYSTEMS

Project Assignment
"Reproducing published RecSys works"

Course instructor: Evgeny Frolov

Completed by:
Anastasiia Demidova
Maksim Bobrin
Nikita Bogdanov
Sergei Kholkin

2023

1 Problem statement

The project aims to replicate the results of published works on Recommender Systems, specifically Collaborative Filtering, using Jax instead of PyTorch. The project will focus on implementing an AutoEncoder model [9] in hyperbolic geometry [8] for Recommender Systems application.

By combining AutoEncoder and hyperbolic geometry, the project aims to develop a more efficient and accurate Recommender System that can handle large and complex datasets. The implementation of this model in JAX will also allow for faster computation and more efficient use of all hardware resources compared to other deep learning frameworks.

Overall, the project's goal is to contribute to the field of Recommender Systems by exploring new techniques and frameworks that can improve the accuracy and efficiency of these systems.

2 MovieLens 1M

In our work, we used such dataset as the MovieLens 1M [2]. This is a popular benchmark dataset used for evaluating Recommender Systems. It contains ratings data from 6,040 users on 3,706 movies, with a total of 1,000,209 ratings. The ratings range from 1 to 5 stars, with half-star increments. The dataset also includes demographic information about the users such as age, gender, and occupation. The data was collected by the GroupLens Research Project at the University of Minnesota and was released in 2003. The dataset has been widely used in research on Recommender Systems and has become a standard benchmark for evaluating the performance of different algorithms.

3 Data Preprocessing

In order to ensure that our Recommender System was trained and evaluated on unbiased data, we performed data preprocessing techniques such as data splitting by timestamp and dropping users and movies from the validation set that were not present in the training set.

Data splitting by timestamp involves dividing the dataset into training and validation sets based on the time at which the ratings were given. This ensures that the validation set contains only ratings that were given after the training set, thereby simulating a real-world scenario where the system is trained on historical data and evaluated on new data.

We used a quantile-based approach to split the data, where we chose the 0.98 quantile as the threshold for the training set. This means that 98% of the ratings were used for training, while the remaining 2% were used for validation.

In addition, we dropped users and movies from the validation set that were not present in the training set. This ensured that our validation set had only users and movies that were already seen in the training set, thereby avoiding any issues of data leakage or bias.

4 Collaborative Filtering

Collaborative Filtering is a method used by recommender systems to make predictions about an interest of a specific user by collecting taste or preferences information from many other users.

The technique of Collaborative Filtering has the underlying assumption that if a user A has the same taste or opinion on an issue as the person B, A is more likely to have B's opinion on a different issue. Its graphical interpretation is shown in the figure 1.

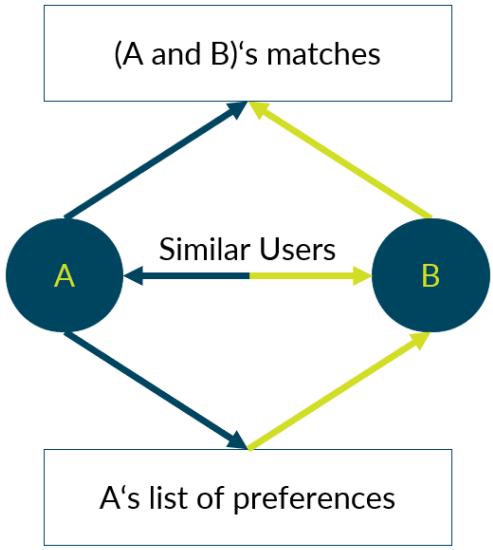

Figure 1: Collaborative Filtering operation scheme

Collaborative Filtering has been found to be effective in making personalized recommendations for users. However, it also has some limitations, such as the cold-start problem, where new users or items have no previous ratings or preferences, and the sparsity problem, where there may be limited overlap between users' preferences. To address these issues, hybrid recommendation systems that combine Collaborative Filtering with other techniques such as Content-based Filtering or Knowledge-based Filtering have been developed. Though, we considered only warm-start scenario in this work, so it was enough to use the standard Collaborative Filtering task.

5 AutoEncoders

Forward pass: given sparse x, compute dense f (x) and loss using equation;
Backward pass: compute gradients and perform weight update;
Predictions: top n indices of AE outputs;
Both for users and items.

$$MMSE = \frac{m_i(r_i - y_i)^2}{\sum_{i=0}^{i=n} m_i}, \tag{1}$$

where $m_i = 1$ if $r_i \neq 0$ else $m_i = 0$.

6 Hyperbolic model (Poincaré)

6.1 Hyperbolic geometry

Hyperbolic geometry is such a geometry with a constant negative curvative, while the Euclidean spaces have a constant positive curvature. This property leads to different geometry properties

of hyperbolic spaces, such as ability to efficiently describe complex networks [7], which is graph of user-item interactions basically is [5]. Hyperbolic geometry has been widely used for various machine learning tasks [10] and particularly for recommender systems [8].

The model of Hyperbolic geometry we will be using is Poincaré ball

$$\mathcal{B}^n = \{x \in \mathbb{R}^n : \|x\| < 1\} \tag{2}$$

equipped with Riemannian metric

$$g^{\mathcal{B}}(x) = \lambda_x^2 g^E(x), \lambda_x = \frac{2}{1 - \|x\|^2}, g^E \text{ - Euclidean metric} \tag{3}$$

so the distance between points on the manifold is as follows:

$$d(x, y) = arcosh(1 + 2\frac{\|x - y\|^2}{(1 - \|x\|^2)(1 - \|y\|^2)}) \tag{4}$$

As you can see conformal factor 3 grows hyperbolically and tends to infinity near the boundaries and difference between two quite close be Euclidean metric g^E points can tend to infinity near the boundaries 4. Also volume grows exponentially when approaching the boundaries of the ball. This are the main differencies between Euclidean geometry and Hyperbolic geometry which suppose to help us with describing user-item relations.

6.2 Operations of Poincaré ball

Addition operation can be defined through Gyrovector space [12]:

$$x \oplus y = \frac{(1 + 2\langle x, y\rangle + \|y\|^2)x + (1 - \|x\|^2)y}{1 + 2\langle x, y\rangle + \|x\|^2\|y\|^2} \tag{5}$$

The other operations can be defined through exploiting tangent space. Riemannian manifolds $\mathcal{M}$ at each point have a tangent space $\mathcal{T}_x\mathcal{M}$ which is isometric to Euclidean space and known Euclidean space operations such as multiplication on scalar $c \otimes x$ can be generalized through mapping onto $\mathcal{T}_x\mathcal{M}$, performing Euclidean operation there and then mapping back onto $\mathcal{M}$.

Operation of projection from manifold to tangent space $Exp_x(v) : \mathcal{M} \to \mathcal{T}_x\mathcal{M}$:

$$Exp_x(v) = x \oplus \left(\tanh\left(\frac{\lambda_x\|v\|}{2}\right)\frac{v}{\|v\|}\right) \tag{6}$$

Operation of projection from tangent space onto manifold $Log_x(v) : \mathcal{T}_x\mathcal{M} \to \mathcal{M}$

$$Log_x(y) = \frac{2}{\lambda_x}\tanh^{-1}\left(\|-x \oplus y\|\right)\frac{-x \oplus y}{\|-x \oplus y\|}, \tag{7}$$

So all the other operations (for example activation function) with Euclidean analog f^E on Poincaré ball can be defined as follows $f_x^{\mathcal{B}} : \mathcal{M} \to \mathcal{M}$:

$$f_x^{\mathcal{B}} = Exp_x \circ f^E \circ Log_x : \mathcal{M} \to \mathcal{M} \tag{8}$$

For example crucial for neural networks matvec:

$$M^{\otimes}(x) = \tanh(\frac{\|Mx\|}{\|x\|}\tanh^{-1}(\|x\|))\frac{\|Mx\|}{\|x\|} \tag{9}$$

6.3 Neural Networks on Poincaré ball

Using all the operation showed above such as addition 5, matvec 9 and generalized framework for getting activation functions 8 we can finally derive Multi Layer Perceptron (MLP) with one hidden layer model:

$$MLP^{\mathcal{B}}(x) = M_2^{\oplus}(f_0^{\mathcal{B}}((M_1^{\oplus}(x)) \oplus b_1)) \oplus b_2, \text{where } M_2, M_1\text{- Layer weights}, b_2, b_1 \text{ - Layer biases} \tag{10}$$

We can the extension of this model to several hidden layer to build autoencoder on Poincaré ball.

6.4 Riemannian optimization

Just having a model is not enough since it has to be optimized. Traditionally neural networks are optimized using variations of Stochastic Gradient Descent [11]. For optimization over Poincaré ball manifold we can use Riemannian Stochastic Gradient Descent (RSGD)[3]. The main idea of algorithm is as follows:

- Obtain Euclidean gradient by differentiating of Euclidean notions of operation on the manifold (ex. 10, 5)
- Project Euclidean gradient onto tangent space $\mathcal{T}_x\mathcal{M}$
- Perform update on $\mathcal{T}_x\mathcal{M}$ using Euclidean addition and scalar multiplication
- Project result back onto manifold $\mathcal{M}$

RSGD algorithm is implemented in geoopt PyTorch library [6] and also there is growing library for Riemannian optimization on Jax [4] called Rieoptax [13] but implementation is rather incomplete. So we had to implement it ourselves. That was done and tested on Linear Model (which you can check [1])

6.5 AutoEncoder on Poincaré ball

Unfortunately something has gone wrong and we could properly optimize MLP on Poincaré ball with one or several hidden layers and couldn't build an Hyperbolic Autoencoder...

7 AutoEncoder results by Hit Rate

As a practical result, we were only able to implement autoencoder using Jax instead of PyTorch. Also, we used the Hit Rate at top-10 as a metric to evaluate the performance of their autoencoder model, as shown in the figure 2.

8 Conclusion

Overall, we considered and explore already existing Recommender System work related with Collaborative Filtering task. Unfortunately, we did not achieve all mentioned goals. Because there were issues with implementing hyperbolic geometry for our AutoEncoder model.

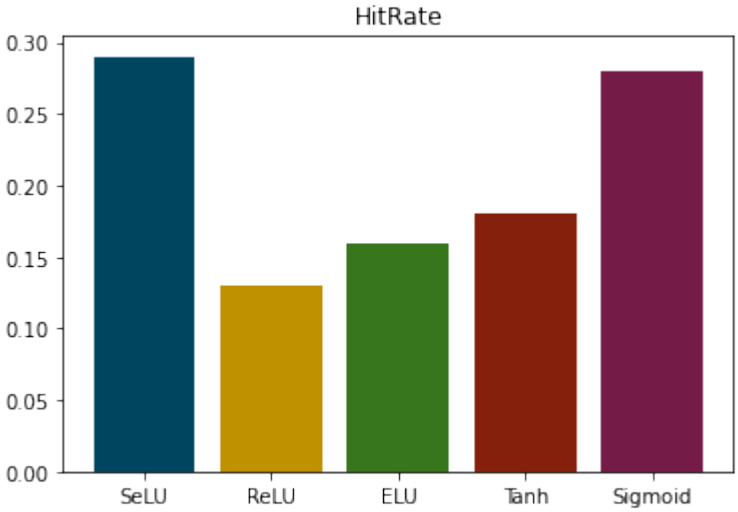

Figure 2: HR@10 of AutoEncoder

As you may find in our github repository [1], the following objectives have been reached:

1. Implemented AutoEncoder model;
2. Applied simple Poincare ball Neural Networks on toy examples;
3. Riemannian optimization for poincare ball NN was also applied;
4. Reached acceptable score for AutoEncoder on movielens 1M.

Taking into account results obtained in the course of this work, there is a future opportunity in the full implementation of a working a hyperbolic autoencoder model.

References

[1] Github repository. `https://github.com/SKholkin/Poincare_AE_recsys`. Accessed: 2023-03-24.

[2] MovieLens. GroupLens. `https://grouplens.org/datasets/movielens/`. Accessed: 2023-03-20.

[3] Silvére Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 58:2217–2229, 2011.

[4] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[5] Jean-Loup Guillaume and Matthieu Latapy. Bipartite graphs as models of complex networks. In *Combinatorial and Algorithmic Aspects of Networking*, 2003.

[6] Max Kochurov, Rasul Karimov, and Sergei Kozlukov. Geoopt: Riemannian optimization in pytorch. *ArXiv*, abs/2005.02819, 2020.

[7] Dmitri V. Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguñá. Hyperbolic geometry of complex networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 82 3 Pt 2:036106, 2010.

[8] V. Khrulkov I. Oseledets A. Tuzhilin L. Mirvakhabova, E. Frolov. Performance of hyperbolic geometry models on top-n recommendation tasks. `https://arxiv.org/pdf/2008.06716.pdf`, 2020.

[9] B. Ginsburg O. Kuchaiev. Training deep autoencoders for collaborative filtering. `https://arxiv.org/pdf/1708.01715v3.pdf`, 2017.

[10] Wei Peng, Tuomas Varanka, Abdelrahman Mostafa, Henglin Shi, and Guoying Zhao. Hyperbolic deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44:10023–10044, 2021.

[11] Sebastian Ruder. An overview of gradient descent optimization algorithms. *ArXiv*, abs/1609.04747, 2016.

[12] Abraham Albert Ungar. Hyperbolic trigonometry and its application in the poincaré ball model of hyperbolic geometry. *Computers & Mathematics With Applications*, 41:135–147, 2001.

[13] Saiteja Utpala, Andi Han, Pratik Jawanpuria, and Bamdev Mishra. Rieoptax: Riemannian optimization in jax. *ArXiv*, abs/2210.04840, 2022.