# Collaborative Filtering and Optimal Transport for Recommender Systems

Ellington Kirby, Mehdi Inane, and Jules Merigot[1]

[1]PSL Research University - Data Science Project

October 2023

# 1 Introduction

**Problem Statement:** We have a sparse matrix with data of user ratings of items ranging from 0.5 to 5, in increments of 0.5, and must employ to Collaborative Filtering methods as well as alternate methods to predict the unknown values. For this task, we implemented Matrix Factorization first with Gradient Descent and then with Alternating Least Squares. Following this, we were interested in seeing how an Optimal Transport algorithm would perform on the sparse data.

# 2 Matrix Factorization and Results

We implemented basic matrix factorization with gradient descent using the equations seen in class, to serve as a baseline compared to our other methods. The below table shows the results from the four different methods we implemented.

| Method | RMSE Train | RMSE Test | Time |
|---|---|---|---|
| Gradient Descent | 0.89 | 0.98 | 16.46s |
| ALS | 0.77 | 0.88 | 25.62 |
| IOT | 1.16 | 1.15 | 2.1s |
| RIOT | 2.45 | 2.60 | 534s |

Table 1: RMSE and Time Results of our Four Methods

# 3 Alternating Least Squares

Following our gradient descent optimization method, we decided to implement an Alternating Least Squares (ALS) optimization. In situations of sparse data, such as ours, ALS performs better than gradient descent or biased stochastic descent. Knowing this, we implemented an ALS algorithm to improve our Matrix Factorization.

## 3.1 The ALS Algorithm

For our ALS implementation, we used the algorithm found in [1]. We first randomly initialized our user matrix X and our item matrix Y that we utilized for matrix factorization. We chose to do this to start the optimization from different points in the parameter space, reducing the risk of consistently converging to the same poor local minimum.

Since our objective is non-convex, ALS proceeds by fixing the set of variables in our user matrix X as constants, which makes the objective a convex function of our item matrix Y, and thus makes it easier to optimizes. The algorithm works by fixing Y and optimizing X, then fixing X and optimizing Y, and so on until convergence. Our implementation relies on the below formulas used to optimize our user and item matrices:

$$x_u = \left( \sum_{r_{ui} \in r_{u*}} y_i y_i^T + \lambda I_k \right)^{-1} \sum_{r_{ui} \in r_{u*}} r_{ui} y_i \quad ; \quad y_i = \left( \sum_{r_{ui} \in r_{*i}} x_u x_u^T + \lambda I_k \right)^{-1} \sum_{r_{ui} \in r_{*i}} r_{ui} x_u$$

The algorithm runs for a set number of iterations that will be discussed further below. However, we also check for convergence after each iteration so that the algorithm stops early if it has already converged. To do this, we compute the predicted matrix which is a product of our X matrix and the transpose of our Y matrix. This outputs a difference matrix that contains the difference between the

observed data and the prediction, but only for the observed entries, which is done by multiplying the original difference by a binary matrix with the same shape as our input data, where each entry of Omega is 1 if the corresponding entry of the input data is greater than 0, and 0 otherwise. We then take the Frobenius norm of the difference which gives us the error for that iteration. If the difference in error between the current iteration and the previous iteration is smaller than our tolerance parameter, the algorithm stops early.

We unfortunately found that this ALS algorithm was outputting a matrix of predicted rating values with some exceeding the expected limit of 5. To prevent this, we decided to use the ***np.clip()*** function from the numpy package to 'clip' any rating value above 5, and set it to 5 instead. The reasoning behind this is the following: If our algorithm is predicting values above 5, this means that the particular user is predicted to really like that item, and thus would be expected to assign it the maximum rating, which is a 5. Following this logic, we can properly clip any large rating values and set them to 5.

### 3.2 Grid Search

In order to achieve the best possible RMSE with our ALS algorithm, we decided to tune our parameters by using a grid search method. For this, we made our own grid search method that allowed us to test multiple values for our four main parameters: the rank (k) of the user and item matrices X and Y, the number of iterations (iters), the tolerance parameter (tol), and the lambda regularization parameter ($\lambda$). Through our grid search method, we found that the parameters that minimized the RMSE for our ALS algorithm are:

$k = 1, \;\; iters = 125, \;\; tol = 1, \;\; \times 10^{-6}, \;\; \lambda = 0.34$

## 4 Optimal Transport for Collaborative filtering

### 4.1 Optimal transport background

Optimal transport consists in finding the best coupling between two probability distributions. This started with a problem in the military field, when G.Monge investigated the optimal transport plan for building military fortifications using piles of earth. The problem was further developed into a more complex probabilistic setting, and was formulated as the following : Given two marginal distributions $\mu \in \mathbb{R}^m$ and $\nu \in \mathbb{R}^n$, and a cost function $C(x, y)$ describing the cost of moving a unit mass from $\mu_i$ to $\nu_j$, find the best coupling $\pi$ such that, given the set of all possible transport plans :

$$U = \{\pi \in \mathbb{R}^{mn} | \pi \mathbf{1} = \mu, \pi^T \mathbf{1} = \nu\} \tag{1}$$

Define $\pi^* = argmin_{\pi \in U} \mathbb{E}_{x,y \ \pi}[C(x, y)]$, as the optimal transport plan minimizing the total expected cost.

### 4.2 Matching

Optimal transport presented a good approach for solving matching problems [2], such as ecological inference [3], election predictions [4] and migration flow problems [5]. Note that collaborative filtering could be considered as a matching problem, however most traditional matching approaches yield poor results when applied to collaborative filtering due to the absence of supply limit of some of its manifestations, such as movie or music recommendation. Thus, our work chose to investigate the applicability of optimal transport to collaborative filtering in the movie contest, that is free of any supply limit, as previous research enforced the supply limit constraint in its experiments.

### 4.3 Inverse Optimal Transport

Matching problems are an interesting application of optimal transport because they entail a data-driven methodology. In this case, suppose that data is available describing pairings between two populations. This data represents the matching history between the two populations, for instance :

- features of husbands and wives and a list of the couples [2]
- inflow and outflow between countries [5]
- voter profiles [3]

If one has marginal distribution for each population, and a cost function, this problem could be solved using traditional optimal transport solving techniques, such as [6] or [7]. However, our problem only features observed scores, hence our motivation in using inverse optimal transport. Such an approach consists in learning the cost matrix from observing the data, and iteratively updating the cost and transport plan until an optimal transport plan, and an optimal cost matrix is found [2, 4]. This is equivalent to solving the following problem :

$$\min_{C} -\sum_{i=1}^{m}\sum_{j=1}^{n} \hat{\pi_{ij}} \log \pi_{ij} = \min_{C} KL(\hat{\pi}, \pi) \tag{2}$$

where $KL(.,.)$ is the Kullback-Leibler divergence, and $\pi$ is the optimal transport plan associated to cost C. We have decided to investigate two algorithms that attempt to find the optimal cost, from [2, 4] and their applicability to unlimited supply constraints as described by movie recommendation. Furthermore, we decided to transform our problem by considering a kernel representation of C. Given matrices U and V, let $C = f(U^T AV)$, that allow us to consider the difference of user space and movie space. U and V measure create a high level description of respectively users and movies in a lower dimensional space, and the optimization search reduces to finding $A$, given that $f$ is an activation function. This reduces computation time and considers the non-linear potential interactions between features in each space. The used optimization algorithms are derived from transforming the initial optimization sum to a min-max problem. Moreover, in order to avoid sparsity in the optimal, improve its smoothness, a reduce computational costs, Regularized Optimal Transport was considered instead [8]. This transformed the objective function to

$$\min_{A,\mu,\nu} \max_{z,w} -\sum_{i=1}^{m}\sum_{j=1}^{n} \hat{\pi_{ij}} \log \pi_{ij} + \lambda(\hat{\mu}, \hat{\nu}, z, w, \mu, \nu, U, V) \tag{3}$$

where $\lambda$ is an entropy-derived regularization leveraging the inherent structure of the cost matrix and the observed and estimated marginal distribution. We refer the reader to the pseudo-code implementation of the two algorithms we used in [2, 4]. For our problem, we set $\hat{\pi}$ as the normalized ratings matrix, $\hat{\pi}\mathbf{1} = \hat{\mu}$ and $\hat{\pi}^T\mathbf{1} = \hat{\nu}$, and explored different options for matrices U and V.

## 5 Methodological Approach and Analysis

### 5.1 Significant Hyperparameters

The most important hyperparameters were the choice of feature matrices used in both IOT and RIOT. The feature matrices were intended to capture latent information on the source and target distributions, in order to inform the construction of the cost matrix.

These feature matrices were observed in two of the reference papers used in the construction of the optimal transport method. In [4], the authors constructed a vector of features for husbands and wives in a marriage matching problem. The features were chosen a-priori from a set of available measures on husbands and wives and clustered into five types. Each potential match was then represented by the cluster centroid in the feature matrix. [2] had a similar approach on the marriage matching problem, though they clustered on 50 clusters and used the centroids of each cluster on 11 features.

The MovieLens dataset used in the matrix completion task did not contain similar features as the marriage dataset seen in the literature. Two methods of feature engineering were explored:

- Non-Negative Matrix Factorization (NMF): A method with similarities to low-rank matrix factorization with the additional constraint to positive quantities [9].
- User-User and Item-Item Dissimilarity using the Jaccard Index. Two square matrices where each index $i, j$ represents the jaccard similarity of Users or Movies $i$ and $j$.

NMF was chosen because of its readily available implementations on sklearn, fast solving speed, and the positive coefficient constraint increasing the interpretability of the resulting matrices.

The motivation of dissimilarity was from [2], where dissimilarity measures were listed as possible inputs to the RIOT formulation.

Ultimately the large size of the dissimilarity matrices made their use impractical. The algorithmic runtime extended exponentially, and did not improve the ratings results. NMF obtained the highest RMSE with 10 components, but as will be discussed shortly, the validity of these results remains questionable. Future work could evaluate other means of constructing feature matrices.

A separate hyper parameter that was experimented with was the choice of regularization used in the proximal gradident of the IOT method. The L1 and L2 norms were evaluated with minimal difference between either. Given the large number of missing values in the dataset, strictly positive constraints were explored. Again the impact was minimal.

## 5.2 Analysis

While OT has been applied in the literature to problems of matching, the lack of clear results when applied to a collaborative filtering problem resulted in several open questions.

- Which output should be evaluated: the learned Cost Matrix or the Optimal Transport plan?
- How can the outputs of the OT algorithms be interpreted as movie ratings?

The question of cost matrix vs transport plan was evaluated by comparing both outputs for both algorithmic methods. Arguments were made in the interest of both interpretations. From the cost matrix point of view, arguments were made that the movie ratings setting is not "supply limited" and evaluating the earth movers distance would not carry the same meaning as in the supply restricted matching setting. However more support for the optimal transport plan interpretation can be found in the literature, where the learned optimal transport plan was used to then compute optimal matchings.

Overall the cost matrix interpretation resulted in lower RMSE, but as will be seen shortly, these results are likely random in nature.

The second question of how best to interpret the outputs as movie ratings ties directly into the issue of interpreting our RMSE results. Figure 1 shows the sorted outputs of the IOT algorithm, and then the sorted outputs of using this learned cost matrix to perform optimal transport. As

can be seen, both distributions are extremely non linear. The vast majority of values are centered around 0.0 in both instances. There are subtle variations in the values, so it is possible that an algorithm could be developed to uncover the significance of the variations. However, the linear bin based method we developed did not seem to capture any deep representation of movie ratings. The lowest RMSE of 1.15 is likely due to the fact that the results stored in the learned IOT cost matrix map directly to a rating of 4.0. Filling the table with 4.0 ratings is closer to the result of filling the table with the mean rating, and thus explains the lower RMSE.

The extremely sparse mapping of the optimal transport plan was observed in both RIOT and IOT, and the resulting mapped ratings produced tables filled with 0.0 ratings. Again, it is possible a more intelligent rating mapping could improve this result, but we did not have the time to investigate this problem.

The conclusion drawn is that the extreme sparsity of the input ratings matrix results in a poorly posed problem for optimal transport. Examples of sparsity in the literature referred to different measures, such as a lack of symmetry between inputs. This sparse input likely heavily influenced the output. This can be seen in the extremely skewed costs and transport plan, where the large majority of outputs clustered around zero.

In the future more work could be done investigating how to properly handle the sparsity of the input, as well as investigate methods of mapping onto the 0.5-5.0 ratings scale.

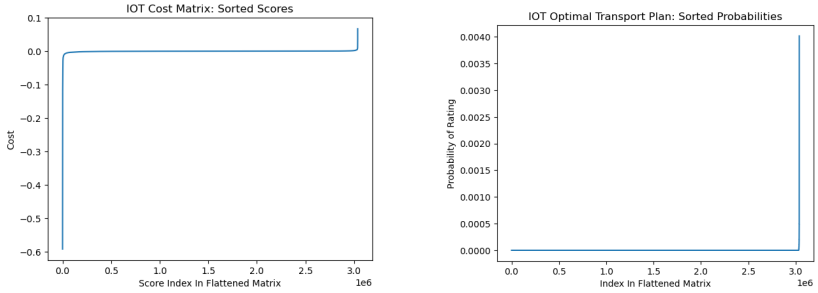


Figure 1: Comparison of sorted outputs of IOT algorithm: costs on the left and the transport plan on the right

# References


[1] R. Zadeh, "Matrix completion via alternating least square (als)," *Stanford CME 323: Distributed Algorithms and Optimization*, Spring 2015.

[2] R. Li, X. Ye, H. Zhou, and H. Zha, "Learning to match via inverse optimal transport," *Journal of machine learning research*, vol. 20, 2019.

[3] B. Muzellec, R. Nock, G. Patrini, and F. Nielsen, "Tsallis regularized optimal transport and ecological inference," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, Feb. 2017.

[4] S. Ma, H. Sun, X. Ye, H. Zha, and H. Zhou, "Learning cost functions for optimal transport," *arXiv preprint arXiv:2002.09650*, 2020.

[5] A. M. Stuart and M.-T. Wolfram, "Inverse optimal transport," *SIAM Journal on Applied Mathematics*, vol. 80, p. 599-619, févr 2020. Funding by NSF.

[6] F. Wang and L. J. Guibas, "Supervised earth mover's distance learning and its computer vision applications," in *Computer Vision – ECCV 2012* (A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, eds.), (Berlin, Heidelberg), pp. 442–455, Springer Berlin Heidelberg, 2012.

[7] M. Cuturi and D. Avis, "Ground metric learning," *Journal of Machine Learning Research*, vol. 15, no. 17, pp. 533–564, 2014.

[8] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport," in *Advances in Neural Information Processing Systems* (C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, eds.), vol. 26, Curran Associates, Inc., 2013.

[9] D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *Advances in neural information processing systems*, vol. 13, 2000.