Optimal Transport for Recommendation Systems

Eloise Zaghrini, Matthieu Rolland, Nathan Bigaud, Mateusz Pyla

Github: Bigaud Rolland Zaghrini

Data Science Project

Master IASD, Université Paris Dauphine-PSL

.

ABSTRACT

In this paper we summarize our attempts to use the Optimal Transport to build a movie recommendation system. We present the Optimal Transport (OT) problem and the Inverse Optimal Transport (IOT) problem on the synthetic data. We show our work on the implementation of four different methodologies to use OT and IOT for the matching problems, focusing for the most part on synthetic datasets rather than the movielens dataset. We indicate advantages and challenges of various approaches as well as point out the learning outcomes.

Keywords: Optimal Transport, Inverse Problem, Recommendation system

1. INTRODUCTION

1.1 Forward Optimal Transport

The original problem, as stated by Monge, focused on taking a set of points at a certain distance of each other, and finding the permutation of those points requiring the minimal cost in terms of distance travelled. For the sake of simplicity and as it is best suited for movie recommendation, we will focus on the discrete case of optimal transport (OT) problem.

This problem is shown to be intractable, so instead we focus on the Kantorovich' relaxation problem, which allows for the mass of a given point to be split among other points, and doesn't restrict the problem to permutations.

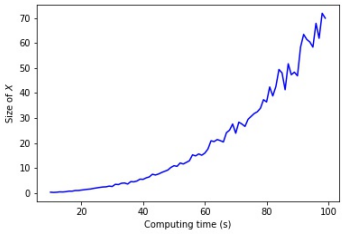

Figure 1. Toy example: computing time for the Kantorovich's problem from $X \in \mathbb{R}^{n\times 2}$ to $Y \in \mathbb{R}^{30\times 2}$

Formal problem definition: The Kantorovich problem is a constrained convex minimization problem. Its objective is $\min_{\mathbf{P}\in\mathbf{U}(\mathbf{a},\mathbf{b})}\langle \mathbf{C}, \mathbf{P}\rangle$, with $C : \mathbb{X} \times \mathbb{Y} \to \mathbb{R}^+$ the cost of transporting one unit of mass from x to y, and $P : \mathbb{X} \to \mathbb{Y}$ transports $a \in \mathcal{P}(\mathbb{X})$ to $b \in \mathcal{P}(\mathbb{Y})$ where $P \in \mathbf{U}(\mathbf{a}, \mathbf{b}) := \{\pi \in \mathbf{R}_+^{m\times n} \quad | \quad \pi\mathbf{1} = \mathbf{a} \quad , \quad \pi^T\mathbf{1} = \mathbf{b}\}$

Linear Programming: In discrete case, computing OT distance amounts to solving a linear programming problem. This is computationally expensive - the best algorithms work with time complexity $\mathcal{O}\left(n^3 \log n\right)$ (see figure 1).

1.2 Entropic Regularization & Sinkhorn algorithm

The idea of entropic regularization of optimal transport is to use the discrete entropy H as a regularizing function to obtain approximate solutions of the original transport problem. H being strictly concave, adding -H with a weight ε makes the problem strictly convex and easier to solve. Note that the amount of entropy is monitored with ε and taking $\varepsilon = 0$ is equivalent to the non-regularized problem. Hence, the Regularized OT problem can be written as:

$$\mathbf{P}\varepsilon = \min_{\mathbf{P}\in\mathbf{U}(\mathbf{a},\mathbf{b})} \langle \mathbf{P}, \mathbf{C}\rangle - \varepsilon\mathbf{H}(\mathbf{P}) \quad with \quad \mathbf{H}(\mathbf{P}) \stackrel{\text{def.}}{=} -\sum_{i,j} \mathbf{P}_{i,j} \left(\log\left(\mathbf{P}_{i,j}\right) - 1\right)$$

The next figure illustrates the effect of entropy to regularize a linear program over the simplex Σ_3 (which can thus be visualized as a triangle in two dimensions). Note how the entropy pushes the original solution to an "entropic center" of the triangle.

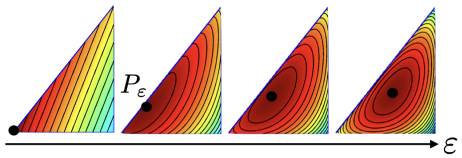

Figure 2. Impact of ε on the optimizationof a linear function on the simplex 3,for a varying ε.

In our practical case, adding some entropy will result in points of distribution a exchanging little amount of mass with several points of distribution b, that means having a very less sparse transport plan matrix P. As a result this enhances the stability of the global solution (for example, the whole transport plan P will not vary too much in case we add few more points in one of the distributions).

We can also show that the solution of the latter problem is unique and has a form $\mathbf{P} = \boldsymbol{diag}(\mathbf{u})\mathbf{K}\boldsymbol{diag}(\mathbf{v})$, where $\mathbf{K} = \exp(-\mathbf{C}/\varepsilon)$ and $(\mathbf{u}, \mathbf{v})$ two scaling variables that must satisfy the mass conservation constraints inherent to $\mathbf{U}(\mathbf{a}, \mathbf{b})$: (i) $\mathbf{P}\mathbf{1}_n = \mathbf{a} \iff \mathbf{u} \odot (\mathbf{K}\mathbf{v}) = \mathbf{a}$ (Row constraint) and (ii) $\mathbf{P}^\top \mathbf{1}_m = \mathbf{b} \iff \mathbf{v} \odot \left(\mathbf{K}^\top \mathbf{u}\right) = \mathbf{b}$ (Column constraint). This whole process can be computed using the Sinkhorn algorithm described in the appendix 5.

1.3 Inverse Problem

Using OT in the forward way, either with linear solvers or Sinkhorn algorithm, allows us to find the optimal relations between users and movies to maximize the global satisfaction of users, assuming the knowledge of the costs between users and movies, as for instance in the third examined.[1]

Considering ratings as transport plan, we could then look at the problem the other way, assuming the cost in unknown. Having empirical distribution of ratings between users and movies, we looked into the optimal cost function between two probability measures given the transportation plan between the spaces.

2. APPROACHES AND RESULTS

2.1 Our process

Initially, our main goal was understanding the basics of OT. Hence, we started by trying to understand the problem from theoretical point of view and spent some time on the linear solvers and regularized solutions for the OT problem in two dimensional space. In order to make sure all members of the team were on the same page, we agreed for this part to be done by all members semi-independently.
We then moved to the main papers, see 2.3 and 2.4. The following days we were to spend on better understanding of the proposed algorithms as well as the implementation. Our final week was focused on the algorithm implementations that were not initially suggested but seem relevant and easier to code see 2.5 and 2.6, as well as code cleanup and writing the project report.

We used different papers using different methods of OT and Inverse OT, in particular: (1) Learning to Match via Inverse Optimal Transport,[2] (2) Learning Cost Function for Optimal Transport[3] (3) PMD: An Optimal Transportation-Based User Distance for Recommender Systems[1] and (3) SISTA: Learning Optimal transport costs under sparsity constraints.[4]

For movies and users similarity, we used different techniques, including standard word embeddings[5] or new embedding space. The latter suffered a lot from the resource limitations.

2.2 Movie recommendation setup

Let us assume that we have an access to potentially incomplete matrix, where each row indicates the users' ratings of a particular movie. Our goal is to predict the incomplete data as well as indicate for each user the next film.

We can highlight the following challenges we spotted early on in the process: (i) the sparsity of the matrix may be too significant, (ii) user ratings may be highly biased or outdated, and (iii) in most cases the cost function (see 1.1) is unknown, however the matching problem is very sensitive to it.

2.3 Learning to Match via Inverse Optimal Transport

This paper presents two algorithms, IOT and Robust IOT. Both aim at computing a cost matrix knowing the empirical transfer matrix between two distributions. The first trick used in this paper is to only look for cost functions that are kernel functions of the distribution, which allow us to reduce our search into a lower dimension matrix used to represent the latent profile associated with users and movies : $C(A) = k(U^T AV)$. We decided to use polynomial kernels in our implementation.

A first way to learn $C(A)$ would then be estimating the parameter through minimizing the KL divergence between the empirical distribution matrix and the one computed with C(A) and the Sinkhorn algorithm. However, it is shown in the paper that relying too heavily on empirical distributions to optimize can very quickly give very bad results if the distributions are noisy or inaccurate.

We thought it best to try to implement the more robust alternative presented later in the paper, RIOT, which optimizes not only on A, but on user and movie distributions too. It gives a more robust result as it allows uncertainty and softens the constraint. However, the algorithm requires the knowledge of user/user and movie/movie similarity matrices.

We did some research in order to figure out a way to compute those matrices, but few papers relied only on collaborative filtering. They tended to resort to tags used for movies, but we thought it best to try to stick to pure collaborative filtering as it was the main idea of the project. In our toy implementation, we computed those by a basic distance calculation. The algorithm we implemented uses an alternating optimization method, and a gradient-like updating of A.

It is quite heavy computationally, as it has a multiple sinkhorn computations and polynomials root searches to do each iterations, and relies on a lot of hyperparameters and pre-known similarity matrices, which is not always accessible. It took some time to get the algorithm to run, and it still needs a lot of tweaking. Possible improvements would be to find a better way to compute similarities between users, and between movies, probably with content filtering, or by optimising on both of those during the iterations.

2.4 Learning Cost Functions for Optimal Transport

The authors present two algorithms, one continuous, one discrete. Given our object of interest, we focused on the discrete setup. We use the data (samples) to find enough generalized cost function which supposedly should work with unseen users' ratings as well as to be naturally aligned with the underlying distributions, (1.3). Authors assume the access to accordingly movie and user space.

2.4.1 The algorithm for cost learning in the discrete regularized IOT

The algorithm relies heavily on the Sinkhorn algorithm 1.2 and works in two steps. This is an iterative process in which we first scale the matrix to obtain $\mathbf{K}$. Then, the second step is a proximal gradient descent: $\mathbf{c} = \text{argmin}_{\mathbf{c}}\{R(\mathbf{c}) + \frac{1}{2\gamma}\|\mathbf{c} - (\varepsilon\log(\mathbf{K}))\|\}$, where $R(\mathbf{c})$ represents the constraints on the cost matrix $\mathbf{c}$ that for instance may arise from the regularization. This term helps us to narrow down the search as well as it allows to find the solution quicker in the same manner as the conjugate gradient descent provided $R(\mathbf{c})$ imposes a constraint to a convex set.

2.4.2 Results

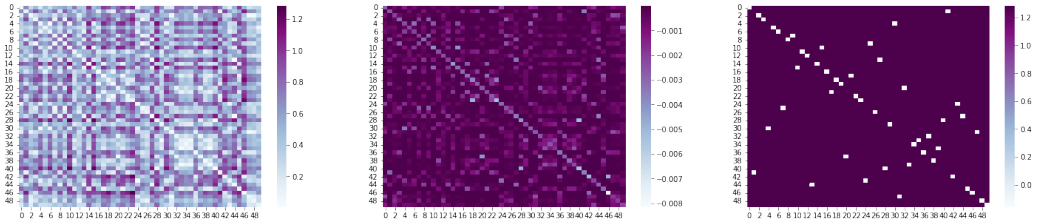

Figure 3. Heat maps of the cost functions for a synthetic 50×50 OT problem. From left: true, regularized, algorithm.

Given the limitations of the resources, we were not able to conducts the experiments on the full movie-user ratings dataset. Instead, we worked around with the synthetic data as well as truncated movie-user dataset.

The biggest advantage of this method is the fact that in comparison to other bi-level optimization methods, the expensive forward OT has to be computed only once. However choosing R function as well as the convex set C on which we impose the constraint remains unclear. Our manual experiments suggest that the solutions have tendency to be simplified.

2.5 PMD: An Optimal Transportation-Based User Distance for Recommender Systems

2.5.1 Approach

This paper focuses on using the Regularized Optimal Transport approach to provide item recommendation in three steps.

1. First by defining a measure of similarity between movies, which can be done either through a pure collaborative filtering approach, relying on user ratings, or by using external information on the movies - the approach chosen by the authors.
2. Next by using the Sinkhorn algorithm to calculate a distance between two users u and v, using the cosine distance matrix as a cost matrix. With $\mathcal{I}_u$ the movies rated by u and $\mathcal{I}_v$ by v:

$$d\left(\mathbf{a_u}, \mathbf{b_v}\right) := \min_{\mathbf{P}_{u,v} \in U(\mathbf{a}_u, \mathbf{b}_v)} \sum_{i \in \mathcal{I}_u} \sum_{j \in \mathcal{I}_v} \mathbf{P}_{u,v}(i,j)\mathbf{C}(i,j) - \epsilon H(P)$$

3. Finally, to predict the ratings of a new user, use the normalized ratings of its K-nearest neighbours, using the distance defined above.

2.5.2 Results

We were able to code the two first steps of the algorithm, and calculate the distance between two users. We used `scikit-learn` tools to calculate a cosine distance movies between based on genres listed in the 'movies' dataset. We then use the `ot` python library, rather than our manually coded Sinkhorn, to ensure efficient calculations. The result seem to show coherent distance between users, as illustrated here: the most overlap between users leads to the shortest distance.

However we were not able to implement the third part of the paper due to computation and time limitation. A described here, the computation of a new user's risk requires to calculate user to user distance for all users - a very expensive calculation. We tried to reduce the size of the dataset using a PCA on users, with limited results. Given more time, we would pursue encouraging results on individual users and implement computation speedups, for instance using HNSW rather than Sinkhorn, to be able to test the algorithm at scale.

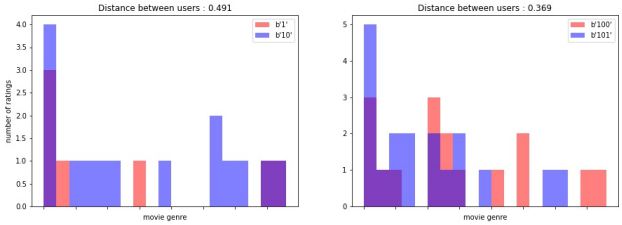

Figure 4. Comparison for two sets of user of movie seen by genre, and associated distance

2.6 SISTA, Learning Optimal transport costs under sparsity constraints

This paper[4] describes a novel iterative procedure called SISTA to learn the cost in optimal transport problems. SISTA is a hybrid between two classical methods, coordinate descent and proximal gradient descent. It alternates between a phase of minimization over the transport potentials and a phase of proximal gradient descent over the parameters of the transport cost. We use the inverse problem of recovering the transport cost $\mathbf{C_{i,j}}$ based on the observed transport plan $\mathbf{P}$.

To achieve this, we assume that the cost function can be represented by a linear combination of basis functions d^k where k is the number of dissimilarity features $d^k_{i,j}$ is some measure of dissimilarity between i and j, and $\beta = (\beta_1, ..., \beta_K)$ is a parameter vector to be learned. Once β is learned using the SISTA algorithm, the cost Matrix can be computed as follows: $\mathbf{c^{\beta}_{ij}} = \sum_{\mathbf{k=1}}^{\mathbf{K}} \beta_{\mathbf{k}} \mathbf{d^k_{ij}}$.

Using this method to our recommendation system, the difficulty comes from computing the dissimilarity matrix $d_{i,j}$ even when starting with k=1. Evaluating the dissimilarity between a user and a movie isn't an easy task when both distributions do not live in the same space. Nevertheless, we could test this algorithm on a very basic simple set up as follows.

We start by creating 2 toy sets X (representing our users) and **Y** (representing our movies) of respectively n and m randomly generated points all living in $\mathbb{R}^2$. Then, we generate a random cost matrix $\mathbf{C_{cost}}$ and compute the optimal transport plan $\mathbf{P_{test}}$ solution of the regularized OT problem. Then with the SISTA algorithm, we calculate the cost matrix $\mathbf{C_{SISTA}}$ from the observed transport plan $\mathbf{P_{test}}$ and see if we are able to retrieve the original cost matrix $\mathbf{C_{cost}}$, which was the case. In figure 5 you will see the two heatmaps of the two matrices.

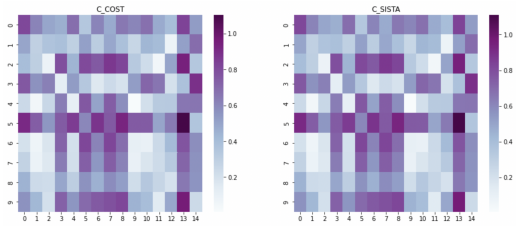

Figure 5. heatmaps of $\mathbf{C_{cost}}$ and $\mathbf{C_{SISTA}}$ for n=10 and m=15, and k=1

3. LEARNING OUTCOMES AND NEXT STEPS

Given the time constraints, we were not able to explore the main papers neatly, however we have very good understanding of the problem formulation, various approaches as well as the biggest challenges. We studied different applications of the problem and we conducted study ablation of the methods.

Despite the fact that it was a group project, we believe it is worth pointing out the individual learning outcomes.

- For Mateusz, it has been really interesting to see how to combine purely theoretical knowledge about the measure theory with the field of data science. The biggest challenge to be faced for him was to adapt the solution[3] to the problem of user-movie recommendation system, in which the distributions live in the different spaces.

- For Eloïse and Matthieu, the project was a great motivation to be curious and to experiment. The subject of OT was completely new, and it was very rewarding to try and figure out ways to transcribe mathematical equations and results into working algorithms.

- For Nathan, this project was a good reminder of the uncertain nature of research - after a few weeks of work we were not able to come to satisfactory results, but gained a much deeper understanding of the issue and are keen to come back to it when possible. It was also a good occasion to note down some 'trick of the trade', such as dealing with unexpected zeros in an algorithm implementation.

4. FURTHER WORK

Given more time, we would :

- Finish implementation of the main papers proposed, continuing our trial and error process until we reach a satisfactory result on the movie-lens dataset

- Look at the the influence on these methods of new data transformations (such as dimensionality reduction on users or more refined approaches to missing data), varying measures of distance between movies (using pure collaborative filtering or more refined distance measure using movie tags), or further hyper-parameters tuning.

- Going beyond the current papers, we would want to further explore Graph Optimal Transport for Cross-Domain Alignment.[6] Solving IOT is also possible with MCMC algorithms with redefining problem to be Bayesian and to sample from the posterior distribution. There are couple of approaches that achieve satisfactory results using for instance Gibbs' Sampling.[7] Another direction that may be taken is to try to focus on the linear approaches towards the optimal transport. There is a very promising paper by Adam Oberman[8] that may be targeting the inefficiency of the linear solvers.

REFERENCES

[1] Carlier, G., Dupuy, A., Galichon, A., and Sun, Y., "Sista: learning optimal transport costs under sparsity constraints," *Advances in Information Retrieval 12036* (2020).

[2] Li, R., Ye, X., Zhou, H., and Zha, H., "Learning to match via inverse optimal transport," (2018).

[3] Sun, H., Zhou, H., Zha, H., and Ye, X., "Learning cost functions for optimal transport," *Journal CoRR* **abs/2002.09650** (2020).

[4] Yitong Meng, Xinyan Dai, X. Y. J. C. W. L. J. G. B. L. G. C., "Pmd: An optimal transportation-based user distance for recommender systems," *arXiv preprint arXiv:2009.08564* (2020).

[5] Řehůřek, R. and Sojka, P., "Software Framework for Topic Modelling with Large Corpora," in [*Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*], 45–50, ELRA, Valletta, Malta (May 2010).

[6] Chen, L., Gan, Z., Cheng, Y., Li, L., Carin, L., and Liu, J., "Graph optimal transport for cross-domain alignment," in [*Proceedings of the 37th International Conference on Machine Learning*], III, H. D. and Singh, A., eds., *Proceedings of Machine Learning Research* **119**, 1542–1553, PMLR (13–18 Jul 2020).

[7] Stuart, A. M. and Wolfram, M.-T., "Inverse optimal transport," (2019).

[8] Oberman, A. M. and Ruan, Y., "An efficient linear programming method for optimal transportation," (2015).

5. APPENDIX

The procedure of the Sinkhorn-Knopp algorithm:

1. Input: Marginal distributions a, b, cost matrix $\mathbf{C}$, regularization parameter ε
2. Initialize: $\mathbf{K} = \exp(-\mathbf{C}/\varepsilon)$ and $a = \mathbf{1}_n$
3. Repeat until convergent
   - $\mathbf{u} \leftarrow \frac{\mathbf{a}}{\mathbf{K}v}$
   - $\mathbf{v} \leftarrow \frac{\mathbf{b}}{\mathbf{K^T u}}$
4. $\mathbf{P} = \boldsymbol{diag}(\mathbf{u}) \quad \mathbf{K} \quad \boldsymbol{diag}(\mathbf{v})$

The procedure of the SISTA IOT algorithm :

Input: Initial guess of parameter vector β^0, of potentials u^0 and v^0, step size ρ, and dissimilarity measures d_{ij}^k while not converged do (Sinkhorn step). Set $c_{ij}^{\beta^t} := \sum_{k=1}^K \beta_k^t d_{ij}^k$ and update:

$$\begin{cases} \exp\left(u_i^{t+1}\right) = \dfrac{p_i}{\sum_{j=1}^N \exp\left(v_j^t - c_{ij}^{\beta^t}\right)} \\ \exp\left(v_j^{t+1}\right) = \dfrac{q_j}{\sum_{i=1}^N \exp\left(u_i^{t+1} - c_{ij}^{\beta^t}\right)} \end{cases}$$

(ISTA step). Let $\pi_{ij}^{\beta^t} := \exp\left(u_i^{t+1} + v_j^{t+1} - c_{ij}^{\beta^t}\right)$. For $k = 1, \ldots, K$

$$\beta_k^{t+1} = \mathrm{prox}_{\rho\gamma|\cdot|}\left(\beta_k^t - \rho \sum_{1 \le i,j \le N} \left(\hat{\pi}_{ij} - \pi_{ij}^{\beta^t}\right) d_{ij}^k\right)$$

end while Return: β

$$with \quad \mathrm{prox}_{\rho\gamma|\cdot|}(z) = \begin{cases} z - \rho\gamma & \text{if } z > \rho\gamma \\ 0 & \text{if } |z| \le \rho\gamma \\ z + \rho\gamma & \text{if } z < -\rho\gamma \end{cases}$$

The procedure of the RIOT algorithm :

Input: observed matching matrix $\hat{\pi}$, cost matrices C_u, C_v, regularization parameter $\lambda, \lambda_u, \lambda_v$

for $l = 1, 2, \cdots, L$ do

$Z \leftarrow \exp(-\lambda C)$

$M \leftarrow \delta\left(\boldsymbol{z}\mathbf{1}^T + \mathbf{1}\boldsymbol{w}^T\right) \odot Z$

Initialize $\boldsymbol{\xi}^{(0)}, \boldsymbol{\eta}^{(0)}$

for $k = 1, 2, \cdots, K$ do

$\theta_1^{(k)} \leftarrow$ root of $p(\theta)$

$\theta_2^{(k)} \leftarrow$ root of $q(\theta)$

$\boldsymbol{\xi}^{(k)} \leftarrow \frac{\hat{\mu}}{\left(M - \theta_1^{(k)} Z\right)\boldsymbol{\eta}^{(k-1)}}$

$\boldsymbol{\eta}^{(k)} \leftarrow \frac{\hat{\nu}}{\left(M - \theta_2^{(k)} Z\right)^T \boldsymbol{\xi}^{(k)}}$

end for

$\boldsymbol{a} \leftarrow \frac{1}{\lambda} \log \boldsymbol{\xi}^{(k)}, \quad \boldsymbol{b} \leftarrow \frac{1}{\lambda}\boldsymbol{\eta}^{(k)}, \quad \theta = \theta_1^{(k)}$

$\pi \leftarrow \exp\left(\lambda\left(\boldsymbol{a}1^T + \mathbf{1}\boldsymbol{b}^T - C\right)\right)$

$\nabla_A \leftarrow \sum_{i,j=1}^{m,n} \lambda\left[\hat{\pi}_{ij} + (\theta - \delta)(z_i + w_j)\,\pi_{ij}\right] C'_{ij}(A)$

$A \leftarrow A - s\nabla_A$

$\boldsymbol{a}_1 \leftarrow$ Sinkhorn-Knopp $(C_u, \pi\mathbf{1}, \hat{\boldsymbol{\mu}}, \lambda_u)\,[1]$

$\boldsymbol{a}_2 \leftarrow$ Sinkhorn-

$\boldsymbol{z} \leftarrow \frac{1}{\lambda_u} \log \boldsymbol{a}_1, \quad \boldsymbol{w} \leftarrow \frac{1}{\lambda_v} \log \boldsymbol{a}_2$

end for