Low-rank optimal transport and application to color transfer

Benoît Müller *

A project for the course Low-rank Approximation Techniques given by Prof. Daniel kressner

December 22, 2022

Abstract

In this project, we aim to solve an entropy regularized version of the Kantorovich optimal transport problem, by using a low-rank approximation technique on Sinkhorn algorithm. We apply the method to color transfer for images. We obtain newly colored images, and by using the low-rank approximation technique with a rank of only 4% of the dimension, we obtain visually similar images with a time computation reduced by half.

The optimal transport framework

Let's introduce the optimal transport framework. We would like to transport some mass distributed in a space to another, and in a optimal way with respect to a certain cost. The total mass stay the same before and after transportation so it can be supposed to have a value of one. This will allow us to think of the mass distributions on the spaces as probability distributions.[1]

Mass transportation In a discrete and finite setting, let's define the original space $\mathcal{X} = \{x_1, \cdots, x_n\}$ and a target space $\mathcal{Y} = \{y_1, \cdots, y_m\}$. In $\mathcal{X}$, the mass is distributed with respect to a vector $\mathbf{p}$ in the simplex $\Delta_n = \{\mathbf{p} \in \mathbb{R}^n | \sum_i \mathbf{p}_i = 1\}$, such that the mass located in x_i is $\mathbf{p}_i$. Similarly, $\mathcal{Y}$ has a target distribution of q. In the Kantorovich's formulation, the transportation of the mass is defined by a coupling, which indicate how many mass of x_i goes to x_j. This gives us a matrix $P \in \mathbb{R}^{n \times m}$ such that we transport P_{ij} from x_i to x_j. As a result, summing the contributions we get the relations $\mathbf{p}_i = \sum_j P_{ij}$ and $\mathbf{q}_j = \sum_i P_{ij}$ which implies that $1 = \sum_i \mathbf{p}_i = \sum_{i,j} P_{ij}$ so P is a stochastic matrix, and $\mathbf{p}$ and $\mathbf{q}$ can be understood to be the marginal distributions of P.

We will use matrix products $P\mathbb{1}_m = \mathbf{p}$ and $\mathbb{1}_n^\top P = \mathbf{q}$ where $\mathbb{1}_k \in \mathbb{R}^k$ is the vector with 1 in all entries. We define the space of couplings between $\mathbf{p}$ and $\mathbf{q}$ as

$$\mathcal{M}(\mathbf{p}, \mathbf{q}) = \{P \in \mathbb{R}_+^{n \times m} | P\mathbb{1}_m = \mathbf{p}, \mathbb{1}_n^\top P = \mathbf{q}\}.$$

This set is defined by linear constraints and coordinates are bounded in $[0, 1]$, so it is a convex polytope.

*Code written in collaboration with Armelle Hours

[1] A reader already familiar with discrete optimal transport and entropy may want to continue his reading in the *next section* after have seen the *recap of notations* detailed in the implementation section.

Cost minimization We consider that transporting mass form a point to one other induce a cost. We define this cost with a cost function $c : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ so that the cost of transporting mass from x_i to y_j is $c(x_i, y_j)$. We do not necessarily ask the cost to be positive or symmetric, but it is often a function of $x_i - y_j$ or $\|x_i - y_j\|$, for a selected norm. To obtain the total cost associated to a coupling, we sum up all transportation cost, according to their weight in P: $\sum_{i,j} c(x_i, y_j)P_{ij}$. We stock the values of c in a matrix $C_{ij} = c(x_i, y_j)$ so that the cost of a coupling can be written with the Frobenius inner product $\langle ., . \rangle$: $\sum_{i,j} C_{ij}P_{ij} = \langle C, P \rangle$. Notice that the cost is linear in the coupling.

Among all possible couplings, we want to try minimizing the cost, and this define an optimal problem, the Kantorovich problem on discrete finite measures:

$$(KP) = \min_{P \in \mathcal{M}(\mathbf{p},\mathbf{q})} \langle C, P \rangle$$

Note that the minimum exist since $\mathcal{M}(\mathbf{p}, \mathbf{q})$ is compact and the objective function is continuous. This is also a linear program since $\mathcal{M}(\mathbf{p}, \mathbf{q})$ is a convex polytope and the objective function is linear. The problem can hence be solved by linear programming methods, but these are very wide general methods and can be inefficient for high dimensions, and the particular structure of the polytope can be used to develop specialized and maybe better methods.

Entropy regularization We are interested here in the entropic regularization of the Kantorovich problem. This is a reformulation of the problem where we subtract to the objective function the Shannon entropy $H(P) = \sum_{i,j} P_{ij} \log \frac{1}{P_{ij}}$, with the convention $0 \log \frac{1}{0} = 0$. It has multiple advantages: first, the Shannon entropy being strongly concave, it makes the objective function strongly convex and the minimizer unique. Also, the interpretation of the Shannon entropy is that it measures the incertitude of a distribution in the sense that the more entropy you have, the less entries of P are near 0 or 1, so you go against sparsity. This results in the fact that the coupling will have many positives values per row and columns and the transport will be the result of multiple ways participation. Lets write the regularized problem explicitly, using a regularization term $\eta > 0$ and a new objective function V_η:

$$\min_{P \in \mathcal{M}(\mathbf{p},\mathbf{q})} \langle C, P \rangle - \eta^{-1} H(P) = \min_{\mathcal{M}(\mathbf{p},\mathbf{q})} V_\eta \tag{1}$$

Relation with matrix scaling and Sinkhorn algorithm

We would like now to obtain information about the analytic solution of the problem, Since it is a constrained convex problem, we would like to make use of the KKT multiplier theorem. This will lead us to a characterization of K, and we present it in the following result, the proposition 4.3 of [1].

Proposition 1. *The solution of the entropy relaxed optimal problem (1) exists, is unique, and its solution is has the form*

$$P = D_1 K D_2$$

for positives diagonal matrices $D_1, D_2 \in \mathbb{R}^{n \times m}$ and the gibbs kernel $K = e^{-\eta C}$.

Proof. The existence of a solution come from the compacity of $\mathcal{M}(\mathbf{p}, \mathbf{q})$ and the continuity of V_η. Its unicity is a consequence of the η^{-1}-strong convexity of V_η and the convexity of $\mathcal{M}(\mathbf{p}, \mathbf{q})$. Let's first show the strong convexity, we compute the partial derivatives

$$\partial_{P_{ij}} V_\eta(P) = C_{ij} + \eta^{-1}(\log P_{ij} + 1).$$

This is still continuously differentiable and we get that the second derivatives are zero except for the diagonal

$$\partial^2_{P_{ij}} V_\eta(P) = \eta^{-1} \frac{1}{P_{ij}} > \eta^{-1}.$$

This define a positive definite hessian with eigenvalues bigger than η^{-1}, so V_η is indeed η^{-1}-strongly convex. As a result, the minimum is unique, since two different minimizers would define a segment inside the convex feasible set. Its image would be strictly convex by the strict convexity and hence contradicting the minimality of the minimizers.

Since (1) is a convex continuously differentiable optimization problem with linear equality constrains, linearity constraint qualification holds everywhere, strong duality holds and the KKT conditions are necessary and sufficient. The Lagrange function writes

$$L(P, \mu, \nu) = \langle C, P \rangle - \eta^{-1} H(P) - \mu^\top (P\mathbb{1} - \mathbf{p}) - \nu^\top (\mathbb{1}^\top P - \mathbf{q})$$

its partial derivative with respect to P_{ij} is

$$\partial_{P_{ij}} L(P, \mu, \nu) = C_{ij} + \eta^{-1} \log P_{ij} - \mu_i - \nu_j$$

and if we set it to be equal to zero we get

$$\log P_{ij} = \eta\mu_i + \eta\nu_j - \eta C$$

$$P_{ij} = e^{\eta\mu_i} e^{-\eta C_{ij}} e^{\eta\nu_j}$$

In other words,

$$P = \operatorname{diag}(e^{\eta\mu}) e^{-\eta C} \operatorname{diag}(e^{\eta\nu}) =: D_1 K D_2$$

with element-wise exponentiation. □

Matrix scaling an Sinkhorn algorithm The last result shows us that we can search for a coupling of the form $P = D_1 K D_2$ with positive diagonal matrices D_1, D_2. When we apply the marginal constraints, we get $D_1 K D_2 \mathbb{1} = \mathbf{p}$ and $\mathbb{1}^\top D_1 K D_2 = \mathbf{q}$, which can be rewritten $u \odot (Kv) = \mathbf{p}$ and $v \odot (K^\top v) = \mathbf{q}$ with $\odot$ the element-wise product. This is the well known matrix scaling problem, and can be resolved by the Sinkhorn algorithm: initialize values of u and v, and then iteratively update them so that they satisfy one of the equations:

$$u \leftarrow \frac{\mathbf{p}}{Kv} \tag{2}$$

$$v \leftarrow \frac{\mathbf{q}}{K^\top u} \tag{3}$$

(with element-wise division). We stop when we have detected convergence.

Low-rank factorization technique

Let's look at the computational cost of the Sinkhorn algorithm. Updates (2) computes a matrix product of mn operations and then n division operations. This gives $mn + n = n(m+1)$ operations and we get similarly $(n+1)m$ operations for (3). In total, we get

$$2mn + n + m \in \mathcal{O}_{m,n}(mn)$$

per iteration so we see that the matrix vector computation is responsible for most of the time complexity. This motivates the use of a low-rank approximation of K and $K^\top$, since using wisely

a r-rank approximation takes the time complexity of the matrix-vector products to $rm + rn = r(m+n) \in \mathcal{O}_{m,n}(m+n)$, we get a total time complexity of

$$2r(m+n) + m + n = (2r+1)(m+n) \in \mathcal{O}_{m,n}(m+n).$$

Notice that the time complexity for fixed m and n is then proportional to r.

Explicitly, for a r-rank factorization $K \approx AB$, we compute Kv with $A(Bv)$ and $K^\top u$ with $B^\top(A^\top u)$. If r is at least $\text{rank}(K)$, the best approximation given by truncated singular value decomposition (SVD) is actually exact. If not, then the approximation has a certain error, for the truncated SVD its value would be the $k+1$-th singular value of K for the spectral norm. The question is then how does this projection affect the computed solution of our original optimal transport problem? To answer this question, we first clearly state the Sinkhorn algorithm, and then we will present a theorem that links the error of the approximation of K with the optimally of the computed solution in the context of the original problem.

Implementation of Sinkhorn algorithm

For our purpose, we will only use square matrices so let's suppose $n = m$ for the suite and the implementation. We do here a recap of the notions and notations what we will reuse in the algorithm and theorem:

- $\Delta_{n\times m} = \{P \in \mathbb{R}_{\geq 0}^{n\times m} | \mathbb{1}_n^\top P \mathbb{1}_m = 1\}$, the simplex of the doubly stochastic matrices. We write $\Delta_n := \Delta_{n\times 1}$.
- $\mathbf{p}, \mathbf{q} \in \Delta_n$ are the target marginals, stochastic vectors of dimension n.
- $\mathcal{M}(\mathbf{p}, \mathbf{q})$: the feasible space of doubly stochastic matrices that are couplings of $\mathbf{p}$ and $\mathbf{q}$(they have $\mathbf{p}$ and $\mathbf{q}$ as marginals).
- $H : \Delta_{n\times n} \to [0,1]$ the entropy of a coupling, $H(P) = \sum_{i,j} P_{ij} \log \frac{1}{P_{ij}}$ extend by continuity with the convention $0 \log \frac{1}{0} = 0$.
- $V_C : \mathcal{M}(\mathbf{p}, \mathbf{q}) \to \mathbb{R}$ the objective function $V_C(P) = \langle C, P\rangle - \eta^{-1} H(P)$, with $\eta > 0$ the regularization coefficient.
- $\Pi^{\mathcal{S}} : \mathbb{R}_{\geq 0}^{n\times n} \to \mathcal{M}(\mathbf{p}, \mathbf{q})$ the Sinkhorn projector. When K is the Gibbs kernel, it solves the problem $V_C(\Pi^{\mathcal{S}}(K)) = \min_{\mathcal{M}(\mathbf{p},\mathbf{q})} V_C$ and can always be written $\Pi^{\mathcal{S}}(K) = D_1 K D_2$ for some diagonals matrices D_1 and D_2 .
- $P^\eta = \text{argmin}_{\mathcal{M}(\mathbf{p},\mathbf{q})} V_C$ the solution of the problem.

We will estimate the error induced by our approximations. We are doing two of them, we approximate the kernel by a low-rank technique, and then we estimate its Sinkhorn projection with the Sinkhorn algorithm. We are going to state and prove a result from [1] that gives the needed tolerance in term of the wanted accuracy.

Suppose we have an approximation $\tilde{K}$ of K, marginals $\mathbf{p}$ and $\mathbf{q}$. The approximated cost associated to $\tilde{K}$ is $\tilde{C} = -\eta^{-1} \log(\tilde{K})$. The algorithm find some D_1 and D_2, which allow us to define the associated matrix $\tilde{P} = D_1 \tilde{K} D_2$ and cost $\hat{W} = V_{\tilde{C}}(\tilde{P})$.
The Sinkhorn algorithm takes as inputs an approximation $\tilde{K}$ of the Gibbs kernel, and a tolerance δ for the stopping criterion. As precised in [1], the sum of marginal errors is a natural good criterion to detect convergence. Note that K is used in functional matrix-vector form, as well as $K^\top$, for

example using a low-rank factorization. Also, however D_1 and D_2 are diagonal matrices, they are never stored like so but only their diagonal $u = \operatorname{diag} D_1$, $v = \operatorname{diag} D_2$, and the computations are always done in this form. The outputs are the matrices D_1, D_2, and the associated cost $\hat{W}$, computed in a efficient way, see Lemma A of [8] for the proof of the equivalence, but notice the error that a multiplication by η is missing. Since it only multiply the cost by a constant factor, we are still able to use it to compare the optimality of the computed solutions.

Here is the algorithm[2]:

Algorithm 1: Sinkhorn
Compute approximate rescaling for the Sinkhorn projection, and the associated cost.

Input : $\tilde{K} \in R_{>0}^{n\times n}$ (in factorized form), $\mathbf{p}$,$\mathbf{q} \in \Delta_n, \delta > 0$
Output: Positive diagonal matrices $D_1, D_2 \in \mathbb{R}^{n\times n}$, cost $\hat{W}$

$D_1, D_2 \leftarrow I_{n\times n}$
$k \leftarrow 0$
while $\|D_1\tilde{K}D_2\mathbb{1} - p'\|_1 + \|(D_1\tilde{K}D_2)^\top\mathbb{1} - q'\|_1 \geq \frac{\delta}{2}$ **do**
  $k \leftarrow k + 1$
  if *k is odd* **then**
    $D_1 \leftarrow \mathbf{p}'/\tilde{K}D_2\mathbb{1}$
  else
    $D_2 \leftarrow \mathbf{q}'/(D_1\tilde{K})^\top\mathbb{1}$
  end if
end while
$\tilde{P} \leftarrow D_1\tilde{K}D_2$
$\hat{W} \leftarrow \sum_{i=1}^n \log(D_1)_{ii}(\tilde{P}\mathbb{1})_i + \eta^{-1}\sum_{j=1}^n \log(D_2)_{jj}(\tilde{P}^\top\mathbb{1})_j$
return $D_1, D_2, \hat{W}$

We implement the Sinkhorn algorithm in Python, by using a functional representation of K and $K^\top$. This allows the user of the function to decide how to perform matrix-vector multiplication: classically, via pre-computed low-rank factorization or any other sophisticated way. The code is available in a GitHub repository [6].

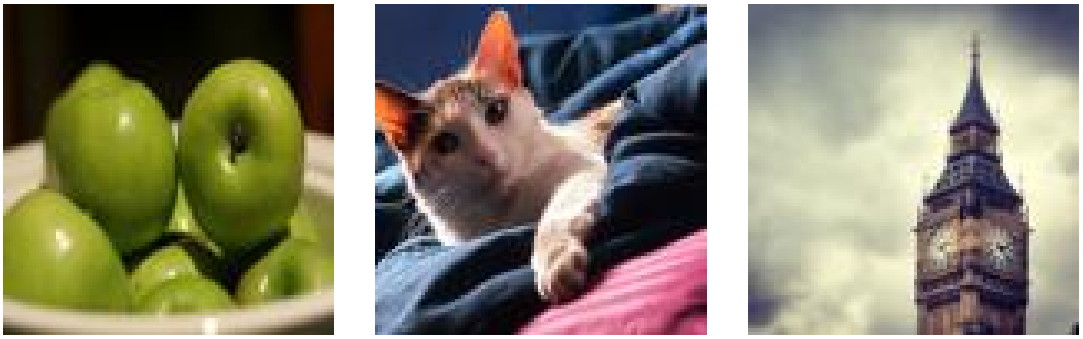
Application to color transfer

The model We apply it to color transfer between images with the following interpretations. The spaces $\mathcal{X} = \mathcal{Y} = \{1, \dots, n\}$ represent the ordered pixels of the images. The distributions $\mathbf{p} = \mathbf{q} = \mathbb{1}_n/n$ are uniform, so all pixel represent the same mass of color. A pixel i of the source image has color $x_i \in [0, 1]^3$ represented in RGB value, as well for the the colors y_j of the target image. This induce a cost $C_{ij} = \|x_i - y_j\|_2^2$ using euclidean distance. A target pixel color can then be changed by taking the weighted average of the source pixel colors, according to the mass transported to this pixel given by the coupling. Explicitly, the weight are the column of the pixel:

$$\tilde{y}_j = \frac{1}{\sum_i P_{ij}} \sum_i P_{ij} x_i$$

[2] correspond to Algorithm 3 in [1].
Pay attention however to the typo in the stopping criterion: $\leq$ should be $\geq$.

Notice the normalization factor so that we get indeed a convex combination that stay in the space $\mathcal{Y}$. The original images are presented in Figure [1], with two different sources and one target.



(a) Source image: Apples (b) Source image: Cat (c) Target image: Big Ben

Figure 1: The original images, of size $n = 100 \times 100$

The computation We set $\eta = 15$. Changing η is equivalent to re-scale the cost, up to multiply the objective function by η. So the choice of representation of colors, in $[0, 1]$ or in $[0, 256]$, directly change the meaning of η. The bigger η is, the less strongly convex the problem is, and so the slower the convergence is. We choose then η by hand so that it is the biggest possible, according to our computational resources. We set tolerance at $\delta = 10^{-15}$ and using our Sinkhorn algorithm and Proposition 1, we obtain an approximate coupling. Notice that by symmetry of the problem with respect to the source and the target, taking the transpose of the coupling gives us the coupling for the opposite direction transport, so we don't need to compute it. The resulting images are show in Figure [2].

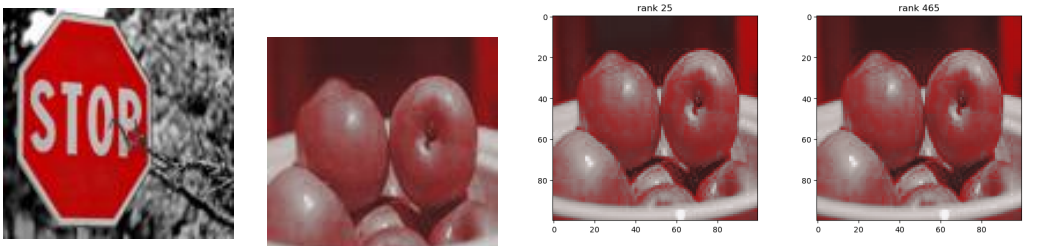
(a) From apples to Big Ben (b) From cat to Big Ben (c) From Big Ben to apples

Figure 2: The new images, of size $n = 100 \times 100$

We see that we can clearly recognize the Big Ben, and that the light distribution is kept, however a little bit uniformly darker (according to the source image). Only the color palette change, accordingly to the palette of the original image. The minimization of the transport cost has for effect to link the colors that are near. The minimization of the entropy has for effect to encourage mixing colors to create the new ones. This effect is traduced by a lower contrast,

Color transfer with low-rank technique

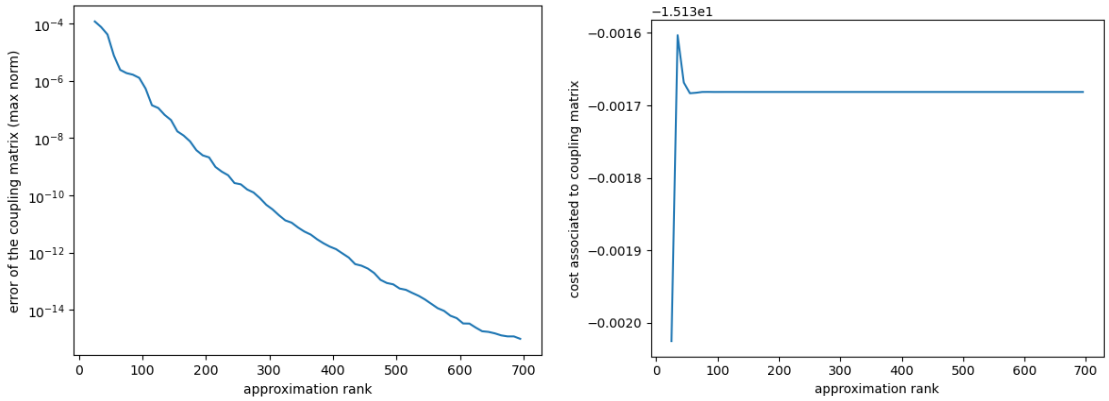
The implementation of the low-rank strategy for the Sinkhorn algorithm comes down to change the functions that compute the matrix-vector multiplications of K and $K^\top$. We use a approximated truncated SVD with the aim of a fast randomized SVD solver from the scikit-learn [5]: sklearn.decomposition.TruncatedSVD[3]. We empirically notice that the rank of the Gibbs kernel K is often very low, in our case it has value 405. This justify even more the use of a low-rank method, since using full rank will give an exact factorization representation of K, and the Sinkhorn algorithm will output the exact same value (supposing the truncation is exact), with the only difference it will be way faster. In practice, the truncation is also an approximation, and the method give a result near to be equal. We use only a new value, the tolerance value $\delta = 10^{-10}$. Resulting images in Figure [3].



(a) Source image: Stop sign (b) With exact kernel (c) Approximaiton with rank 25 (d) Approximaiton with rank 465

Figure 3: New color versions of the apples(Figure [1a])

The Error We see that we cannot observe significant difference. To look numerically the difference, we plot the coupling max-norm error in function of the rank in Figure [4a].



(a) Coupling error in max-norm (b) Coupling cost

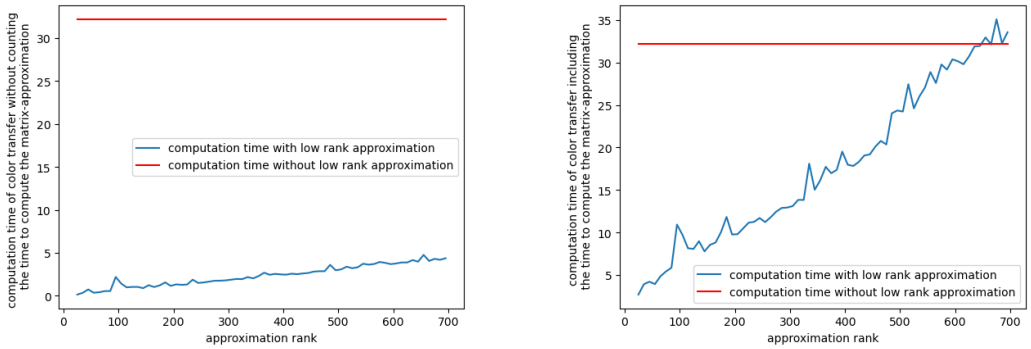
Figure 4: Effects of the low-rank approximation

We see that the error decrease almost exponentially since the y axis is in logarithm scale and that the curve is near to be straight, a bit convex however. At rank 200 (over 10,000), we

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html`

already have an error induced by the low-rank approximation, smaller than the tolerance of the initial computation of the coupling without low-rank approximation. This means that relatively to the precision of the coupling, the low-rank approximation effect becomes negligible. Since Sinkhorn algorithm stopping criterion is based on a marginal error tolerance, changing K change the problem but doesn't change the quality of the computation, so the cost should not change much. It is indeed confirmed by Figure [4b].

The time improvement We can now display the computation time gained by using a low-rank approximations. In Figure [5a], we show the time of computation that the Sinhorn algorithm takes in function of the rank.



(a) Sinkhorn algorithm (b) Low-rank approximation + Sinkhorn algorithm

Figure 5: Time [s] according to rank approximation

We see that indeed the computation is highly faster, six times more at least for the rank plotted. Using the time complexity computed before, to look for the rank where an iteration takes the same time complexity with a matrix or a low-rank factorization, we solve $2mn+n+m = (2r+1)(m+n)$ for $n = m = 100^2$ and get $r = 5,000$. So supposing they all have the same number of iterations [4] , the blue curve should cross the red one around rank 5,000. This is very high in comparison to the values we used and the actual rank 405. However, in a purely application perspective and to be perfectly exhaustive we should take into account the time of computing the low-rank approximation. This is done in Figure [5b]. We see that the low-rank is still faster but the time increase faster and the threshold where the curves cross is taken down to 600. For a full rank approximation, we still have almost half of time computation.

An error bound for low-rank optimal transport

We used a low-rank approximation of the Gibbs kernel, computed an approximation of its Sinkhorn projection, and use it to define a heuristic coupling candidate. By its stopping criterion and its convergence, we know that the Sinkhorn algorithm can give Sinkhorn projection approximation of arbitrary quality. But with the low-rank approximation, we compute the Sinkhorn projection of a approximate kernel, i.e. an approximate formulation of the problem. Is the problem well posed for K in the sense of J. Hadamard?

To understand this, we present the following theorem, corresponding to Theorem 5 of [1]:

[4] It is not the case since a low-rank matrix could have better convergence properties, but we suppose it to get an insight of the sizes numbers we are talking about.

Theorem 1.
If $\tilde{K} \in R_{>0}^{n\times n}$ approximate $K = e^{-\eta C}$ with error

$$\|\log K - \log \tilde{K}\|_\infty \leq \epsilon' = \min\left(1, \frac{\epsilon\eta}{50(\|C\|_\infty \eta + \log\frac{n}{\eta\epsilon})}\right),$$

then using tolerance $\delta = \epsilon'$ and supposed that Algorithm 1 terminate[5]*, it outputs values $D_1, D_2, \hat{W}$ such that*

$$|V_C(P^\eta) - V_C(\tilde{P})| \leq \frac{\epsilon}{2} \tag{4}$$

$$|\hat{W} - V_C(\tilde{P})| \leq \frac{\epsilon}{2} \tag{5}$$

Proof.
Let's proof (4) first. As stated before, the exact optimal solution is the Sinkhorn projection of K in the feasible space: $V_C(P^\eta) = V_C(\Pi^{\mathcal{S}}(K))$ (Corollary 1 in [1]). We decompose the error by triangle inequality and treat each error term one by one:

$$\begin{aligned}
|V_C(\Pi^{\mathcal{S}}(K)) - V_C(\tilde{P})| \leq\ & |V_C(\Pi^{\mathcal{S}}(K)) - V_C(\Pi^{\mathcal{S}}(\tilde{K}))| && (6)\\
&+ |V_C(\Pi^{\mathcal{S}}(\tilde{K})) - V_{\tilde{C}}(\Pi^{\mathcal{S}}(\tilde{K}))| && (7)\\
&+ |V_{\tilde{C}}(\Pi^{\mathcal{S}}(\tilde{K})) - V_{\tilde{C}}(\tilde{P})| && (8)\\
&+ |V_{\tilde{C}}(\tilde{P}) - V_C(\tilde{P})| && (9)
\end{aligned}$$

Term (6): We need to use regularity of $\Pi^{\mathcal{S}}$ and V_C: by Proposition 2 in [1] and hypothesis on $\tilde{K}$,

$$\|\Pi^{\mathcal{S}}(K) - \Pi^{\mathcal{S}}(\tilde{K})\|_1 \leq \|\log K - \log \tilde{K})\|_\infty \leq \epsilon' \leq 1.$$

We get then by Lemma E in [1] that

$$|V_C(\Pi^{\mathcal{S}}(K)) - V_C(\Pi^{\mathcal{S}}(\tilde{K}))| \leq \epsilon'\|C\|_\infty + \eta^{-1}\epsilon' \log(\frac{2n}{\epsilon'}).$$

Term (7) and (9): We need to use the regularity of $C \mapsto V_C$. For all $Q \in \Delta_{n\times n}$ we have

$$\begin{aligned}
|V_C(Q) - V_{\tilde{C}}(Q)| = |\langle C - \tilde{C}, Q\rangle| &\leq \|C - \tilde{C}\|_\infty \|Q\|_1 = \|C - \tilde{C}\|_\infty \\
&= \eta^{-1}\|\log K - \log \tilde{K}\|_\infty \leq \eta^{-1}\epsilon'
\end{aligned}$$

(This is Lemma C in [1]). In particular, taking alternatively $\Pi^{\mathcal{S}}(\tilde{K})$ and $\tilde{P}$ for Q, gives

$$|V_C(\Pi^{\mathcal{S}}(\tilde{K})) - V_{\tilde{C}}(\Pi^{\mathcal{S}}(\tilde{K}))| \leq \eta^{-1}\epsilon'$$

and

$$|V_{\tilde{C}}(\tilde{P}) - V_C(\tilde{P})| \leq \eta^{-1}\epsilon'$$

Term (8): The proposition 3 in [1] measure this error. From the stopping criterion in line 6 of Algorithm 1, $\|\tilde{P}\mathbb{1} - p'\|_1 + \|\tilde{P}^\top\mathbb{1} - q'\|_1 < \frac{\epsilon'}{2} \leq 1$ and $\tilde{P} \in \Delta_{n\times n}$ so

$$|V_{\tilde{C}}(\Pi^{\mathcal{S}}(\tilde{K})) - V_{\tilde{C}}(\tilde{P})| \leq \epsilon'\|\tilde{C}\|_\infty + \eta^{-1}\epsilon' \log(\frac{2n}{\epsilon'})$$

[5] It does, thanks to Theorem 4.2 in [8]

We can now add all bounds together:

$$|V_C(\Pi^{\mathcal{S}}(K)) - V_C(\tilde{P})| \leq \epsilon' \|C\|_\infty + \eta^{-1}\epsilon' \log(\frac{2n}{\epsilon'}) + 2\eta^{-1}\epsilon' + \epsilon' \|\tilde{C}\|_\infty + \eta^{-1}\epsilon' \log(\frac{2n}{\epsilon'})$$
$$= \epsilon'(\|C\|_\infty + \|\tilde{C}\|_\infty + 2\eta^{-1} + 2\eta^{-1}\log(\frac{2n}{\epsilon'})).$$

Since $\|\tilde{C}\|_\infty \leq \|C\|_\infty + \|\tilde{C} - C\|_\infty \leq \|C\|_\infty + \eta^{-1}\epsilon' \leq \|C\|_\infty + \eta^{-1}$, we finally get

$$|V_C(\Pi^{\mathcal{S}}(K)) - V_C(\tilde{P})| \leq \epsilon'(2\|C\|_\infty + 3\eta^{-1} + 2\eta^{-1}\log(\frac{2n}{\epsilon'}))$$

When $\epsilon' = \min\left(1, \frac{\epsilon\eta}{50(\|C\|_\infty \eta + \log\frac{n}{\eta\epsilon})}\right)$ we can show with a bit of algebra (Lemma M in [1]) that this is indeed smaller than $\epsilon/2$.

Now let's show (5). By definition, $V_{\tilde{C}}(\tilde{P}) = \langle C, \tilde{P}\rangle - \eta^{-1}H(\tilde{P})$. The formula for $\hat{W}$ that is in line 15 of Algorithm 1, rewrite this in a more computationally efficient way[6]. The proof of the equivalence is Lemma A in [1]. As a result,

$$|\hat{W} - V_C(\tilde{P})| = |V_{\tilde{C}}(\tilde{P}) - V_C(\tilde{P})| \leq \|\tilde{C} - C\|_\infty \leq \eta^{-1}\epsilon' < \epsilon/2$$

□

Discussion

The results showed that the low rank don't affect much the quality of the transport, and that the computation time is significantly reduced. This method is easy to implement and doesn't disturb the convergence to a precise coupling. We also noticed empirically that the rank of the Gibbs kernel is often very low. This should be quantified and understood better, in order to be able to automatically choose a confident rank without having to do a spectral analysis of the kernel by hand.

Secondly, we explained that the choice of the cost and in particular the color format have impact on the weight of the regularization. What is more it that the coordinate representation of colors change the distances and the notion of proximity in the color space. For example, if we had taken HSL format, (Hue-Saturation-Lightness) the transport would have probably changed. Varying this format could be an interesting question for an image processing perspective.

References

[1] J. Altschuler, F. Bach, A. Rudi, and J. Niles-Weed. "Massively scalable Sinkhorn distances via the Nyström method". In: *arXiv preprint arXiv:1812.05189* (2019). URL: https://arxiv.org/abs/1812.05189.

[2] appliedprobability. *Lagrangian Optimization*. [Online; accessed 9-novembre-2022]. 2017. URL: https://appliedprobability.blog/2017/09/30/lagrangian-optimization/.

[3] Rabin et at. "Adaptive color transfer with relaxed optimal transport". In: *IEEE Int. Conf. Imag. Process* (2014). URL: https://doi.org/10.1109/ICIP.2014.7025983.

[4] Wikipedia contributors. *Entropy (information theory) — Wikipedia, The Free Encyclopedia*. [Online; accessed 9-novembre-2022]. 2022. URL: https://en.wikipedia.org/w/index.php?title=Entropy_(information_theory)&oldid=1120177669.

[6] Pay attention that a typo: the paper misses a factor η. However, this only multiply the cost by a contant factor, so it still allow us to use this formula to compare costs relative order.

[5] David Cournapeau. *scikit-learn*. Version Release 1.2.0. 2022. URL: https://github.com/scikit-learn/scikit-learn/tree/1.2.0.

[6] Armelle hours and Benoît Müller. *Repository: $Low_Rank_Approximation_Techniques_Project$*. Version v2. 2022. URL: https://github.com/Benoit-Muller/Low_Rank_Approximation_Techniques_Project/tree/v2.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[8] G. Peyré and M. Cuturi. "Computational Optimal Transport". In: *arXiv preprint arXiv:1803.00567* (2020). URL: https://doi.org/10.1109/ICIP.2014.7025983.