Homework : the Sinkhorn algorithm

Luca Nenna

February 25, 2022

You may use the language of your choice (Python, Julia, Matlab,...) and return a short report as a jupyter notebook or pdf file (with concise and justified answers, full proofs are not required).**The deadline to submit the report is March 25 at 6 a.m. (New York time).**

The aim of this homework is to solve the following problem by mean of the Sinkhorn algorithm

$$P_\varepsilon = \inf \left\{ \langle \gamma | c \rangle + \varepsilon \mathrm{Ent}(\gamma) \mid \gamma \in \mathbb{R}^{X \times Y},\ \sum_{y \in Y} \gamma_{xy} = \mu_x,\ \sum_{x \in X} \gamma_{xy} = \nu_y \right\},$$

where $\varepsilon > 0$, X and Y are two finite sets with both cardinality N, $\langle \gamma | c \rangle = \sum_{x,y} \gamma_{xy} c(x,y)$, $\mathrm{Ent}(\gamma) = \sum_{x,y} e(\gamma_{xy})$ with

$$e(r) = \begin{cases} r(\log r - 1) & \text{if } r > 0 \\ 0 & \text{if } r = 0 \\ +\infty & \text{if } r < 0 \end{cases}$$

We know from Lecture 2 that the optimal solution to P_ε takes the following form

$$\gamma_{x,y} = \mathrm{diag}(D_\varphi) \exp \frac{-c(x,y)}{\varepsilon} \mathrm{diag}(D_\psi),$$

where $\mathrm{diag}(D_\varphi)$ and $\mathrm{diag}(D_\psi)$ are two diagonal matrices having the diagonal D_φ and D_ψ, respectively. The Sinkhorn algorithm is then defined as

Algorithm 1 Sinkhorn-Knopp algorithm

```
function SINKHORN(μ, ν, K_ε, k_max)
    D_φ^0 ← 1_X,  D_ψ^0 ← 1_Y
    for 0 ⩽ k < k_max do
        D_φ^{k+1} ← μ./(K D_ψ^k)
        D_ψ^{k+1} ← ν./(K^T D_φ^{k+1})
    end for
end function
```

where ./ stand for the element-wise division and $K_{x,y} = e^{\frac{-c(x,y)}{\varepsilon}}$. Here we will always consider $c(x,y) = |x-y|^2$ where $|\cdot|$ denotes the Euclidean norm.

1. Write a function `sinkhorn(mu,nu,Keps,kmax)` that takes as input two vectors `mu` and `nu`, the kernel `Keps` defined as $K_{x,y} = \exp \frac{-(x-y)^2}{\varepsilon}$ and the maximum number of iterations `kmax`. The function will return two vectors `Dphi` and `Dpsi` and a matrix `gamma`.
2. Let X and Y two sets of N random points in $[0,1]$ (i.e in Python one can use the function `np.random.rand`) and consider two measures such that $\mu_x = \frac{1}{N}\ \forall x \in X$ and $\nu_y = \frac{1}{N}\ \forall y \in Y$. Compute the optimal solution to P_ε for different values of ε, plot it and comment the results. What do you expect as $\varepsilon \to 0$?
3. Modify the function `sinkhorn(mu,nu,Keps,kmax)` such that it also returns a vector `err1` containing the constraints satisfaction at each step of the algorithm, that is $err_1^k = \left\|\gamma_1^k - \mu\right\|_1$ where $(\gamma_1^k)_x = \sum_y \gamma_{x,y}^k$. Display `err1` in log-scale by taking the same data as in question 2 and study the impact of ε on the convergence rate of the algorithm.
4. Modify the function by using $err_1^k \leqslant$ `tol` as a stopping criterion. The new function will take as input also a tolerance parameter `tol`: `sinkhorn(mu,nu,Keps,kmax,tol)`.

5. Let μ and ν be the discretization of truncated Gaussian distributions of (mean,variance) $(0.2, 0.1^2)$ and $(0.6, 0.2^2)$ and compute the solution to P_ε by choosing the smallest ε possible. Display the support of the optimal transport plan.
6. Because of the discretization and the regularization, we see that the optimal transport plan is not deterministic. A workaround is to define the *barycentric projection map*

$$T(x) = \frac{\sum_{y\in Y} y\gamma_{x,y}}{\sum_{y\in Y} \gamma_{x,y}}$$

which is well-defined whenever $a_x = \sum_{y\in Y} P_{x,y} > 0$. Use the barycentric projection to compute an approximation of the transport plan between the two measure.