Structured Data: Learning and Prediction
Differentiable Ranks and Sorting using Optimal Transport

Benjamin Cohen, Vladimir Kondratyev, Jean Pachebat

April 22, 2022

Abstract

Sorting is a necessary tool for machine learning, to create algorithms (k-NN) or test-time metrics like top-k classification accuracy or losses based on the rank. Nevertheless it seems to be a difficult task for automatically differentiable pipelines in DL. Sorting gives us two vectors, this application is not differentiable as we are working with integer-valued permutation. In the paper they aim to implement a differentiable proxy of the basic approach.

The article conceive this proxy by thinking of an optimal assignment problem. We sort n values by matching them to a probability measure supported on any increasing family of n target values. Therefore we are considering Optimal Transport (OT) as a relaxation of the basic problem allowing us to extend rank and sort operators using probability measures. The auxiliary measure will be supported on m increasing values with m $\neq$ n. Introducing regularization with an entropic penalty and applying Sinkhorn iterations will allow to gain back differentiable operators. The smooth approximation of rank and sort allow to use the 0/1 loss and the quantile regression loss.

1 Synthesis of the paper

1.1 Assumptions, notations and goal

For clarity we introduce the following notation:

- $O_n \subset R^n$ - the set of increasing vectors of size n.
- $\Sigma_n \subset R_+^n$ - probability simplex.
- 1_n - n-vector of ones.
- Given $c = (c_1, \ldots, c_n) \in R^n$, $\bar{c} = (\mathrm{c}_1 + \cdots + \mathrm{c}_i)_i$.
- Given two permutations $\sigma \in \mathrm{S}_n$, $\tau \in \mathrm{S}_m$ and a matrix $A \in \mathrm{R}^{nm}$, we write $A_{\sigma\tau}$ for the $n \times m$ matrix $[A_{\sigma_i\tau_j}]_{ij}$ obtained by permuting the rows and columns of A using σ and τ.
- $\forall x \in R$, δ_x - Dirac measure on x.
- Lets put ξ being the probability measure, then $\forall \xi \in P(R)$, F_ξ - cumulative distribution function (CDF), and Q_{x_i}: quantile function (generalized if x_i is discrete).

Functions are applied element-wise on vectors or matrices; the operator $\circ$ stands for the element-wise product of vectors.

The article we have studied works mainly with problems of sorting and ranking the array of numbers. The sorting can be defined as a function S:

$$x = (x_1, ..., x_n) \in R^n \xrightarrow{\text{find } \sigma} x_\sigma = (x_{\sigma_1}, ..., x_{\sigma_n})$$

where the array $x_\sigma = (x_{\sigma_1}, ..., x_{\sigma_n})$ is positioned in increasing order. By solving the sorting problem, we obtain two vectors.

- $S(x) := x_\sigma$ - vectors of sorted values

- $R(x) := \sigma^{-1}$ - the rank of each entry of x.

The function R is mainly used in order statistics, k-NN rules or assessing the performance of an algorithm such as 0/1 and top-k classification accuracies. S is used in robust statistics to handle outliers (trimmed, least-quantile regression, median-of-means estimators). The main problems of those functions are that S in not differentiable everywhere and R is piecewise constant (i.e. Jacobian $\frac{\partial R}{\partial x} = 0$ a.e).

1.2 Idea

The article explores idea of replacing usual ranking and sorting operators by differentiable approximations. We could replace the previous test metrics with training losses by replacing R and S by differentiable proxies. We can understand this method if we want to impose constraints on the repartition of samples of a dataset using quantiles. The main contribution of the article relies in the link studied between Optimal Transport (OT) and smoothed operators R, S: indeed we can find the permutation σ of x by solving an Optimal Assignment problem from a measure defined on the support of x to any increasing family y of same length. Optimal assignement problem is defined as

$$x_{\sigma_0} = y_0; ..., x_{\sigma_i} = y_i; , x_{\sigma_n} = y_n;$$

Thus we learn σ of x. To extend OT, we can use target measures with different lengths supports ($m \neq n$) then use OT $\implies$ convex combinations of ranks and sorted values. Problem: Too costly, non-differentiable. To get back differentiable operators, they regularize OT and solve the problem with the Sinkhorn algorithm with complexity $O(nml)$; with l = Card(Iterations for Sinkhorn algorithm convergence) and m the size of the target measure that we can choose small (3 in some experiments).

1.3 Problem to be solved: adapt ranking and sorting as an Optimal Transport Problem, Kantorovich operators

Wasserstein distance between two univariate measures is computed by comparing quantile functions, obtained by inverting CDFs computed with the ordered values of the supports of those measures. Here we pay only $O(nlog(n))$ operations to sort far less than $O(n^3log(n))$ for OT problems.

We will now define ranking and sorting as the solution of an optimal assignment problem between measures supported on R leading to generalized rank and sort operators using the Kantorovich formulation of OT.

OT between 1D measures using sorting
Define two discrete measures $(\xi, \nu) \in (P(R))^2$ with supports respectively x, y and vectors a, b s.t.:

$$\xi = \sum_{i=1}^{n} a_i \delta_{x_i} \text{ and } \nu = \sum_{j=1}^{m} b_j \delta_{y_j}$$

We also introduce a translation invariant, non-negative ground metric:

$$(x, y) \to h(y - x) \text{ with } h : \mathbb{R} \to \mathbb{R}_+$$

Then, the OT problem between ξ and ν is the linear program:

$$OT_h(\xi, \nu) = \min_{P \in U(a,b)} \langle P, C_{xy} \rangle \tag{1}$$

where $U(a, b) := \{P \in {R_+}^{n \times m} | P\mathbf{1}_m = a,\ P^\top \mathbf{1}_m = b\}$.

Write $C_{xy} = [h(y_j - x_i)]_{ij}$. When h is supposed convex, we get a closed form solution using quantile functions:

$$OT_h(\xi, \nu) = \int_0^1 h(Q_\nu(u) - Q_\xi(u))\, \mathrm{d}u \tag{2}$$

The $OT_h(\xi,\ \nu)$ integrates the difference between the quantile functions of $(x_i,\ \nu)$, which is done by inverting F_ξ, F_ν which requires to sort $\mathbf{x}$ and $\mathbf{y}$ to get their sorting permutations σ and τ. 2 recovers the optimal solution P_* in $n + m$ operations. σ and τ then allow us to build the north-west corner solution of 1.

Proposition 1 *Let (σ, τ) be sorting permutations of* $\mathbf{x}$ *and* $\mathbf{y}$ *respectively. Define N to be the north-west corner solution using permuted weights a_σ, b_τ . Then $N_{\sigma^{-1},\tau^{-1}}$ is optimal for 1.*

In that case, $(P_*)_{\sigma,\tau}$ runs from the top-left (north-west) to the bottom right corner. When $n = m$; $a = b = \mathbf{1}_n/n$, $N_{\sigma^{-1},\tau^{-1}}$ is a permutation matrix divided by n and equals to 0 everywhere except for its entries indexed by $(i, \tau(\sigma^{-1}))_i$ equal to $1/n$. It is a solution to the optimal assignment which to the i-th value in $\mathbf{x}$ associates the $(\tau(\sigma^{-1}))_i$-th value in $\mathbf{y}$; i.e. the i-th smallest entry in $\mathbf{x} \rightarrow i$-th smallest entry in $\mathbf{y}$.

Generalizing sorting, CDFs and quantiles using optimal transport Assume $\mathbf{y}$ is sorted, $y_1 \leq ... \leq y_m$. Then $\tau = Id$ and if $n = m$ then the $i - th$ smallest value of $\mathbf{x}$ is assigned to ${\sigma_i}^{-1} \implies R$ and S are written using P_*.

Proposition 2 *Let $n = m$ and $n = m$ and $\mathbf{a} = \mathbf{b} = \mathbf{1}_n/n$. Then for all strictly convex functions h and $\mathbf{y} \in \mathbb{O}_n$, if P_* is an optimal solution to 1, then:*

$$R(\mathbf{x}) = n^2 P_\star \overline{\mathbf{b}} = nP_\star, \quad \begin{bmatrix} 1 \\ \vdots \\ n \end{bmatrix} = nF_\xi(\mathbf{x}), \quad S(\mathbf{x}) = nP_\star^T \quad \mathbf{x} = Q_\xi(\overline{\mathbf{b}}) \in \mathbb{O}_n$$

We can understand this expression as n times CDF of x which are the quantiles of ξ at levels $\bar{b}$. The proposition is only valid when ξ and ν are uniform of same size support. The paper now focuses on more general cases where $m = size(y \leq n$; $\mathbf{a}$ and $\mathbf{b}$ are no longer uniform.

Kantorovich ranks and sorts: compare discrete measures of several sizes and weights. We want to split the weight a_i of x_i to assign it to several y_i ($\Leftrightarrow$ using b_i) $\implies$ the i-th line (or j-th column) of a solution $P_* \in {R_+}^{n*m}$ often has several positive entries. Extending Prop. 2 gives us extended operators named Kantorovich ranking and sorting operators. K-ranking operator $\xrightarrow{calculate}$ convex combinations of rank values; K-sorting operator $\xrightarrow{calculate}$ convex combinations of values contained in x directly. This method is seen as a convex combinations of ranks/values in Euclidean geometry. Future works could use alternative geometries (KL, hyperbolic, etc) on ranks/values. They are both pointwisely quantities and depend on the ordering of $\mathbf{a}$, $\mathbf{x}$, $\mathbf{b}$, $\mathbf{y}$.

Definition 1 *(K-ranks & K-sorts). $\forall\ (\mathbf{x}, \mathbf{a}, \mathbf{y}, \mathbf{b}) \in \mathbb{R}^n \times \Sigma_n \times \mathbb{O}_m \times \Sigma_m$, let $P_\star \in U(\mathbf{a}, \mathbf{b})$ be an optimal solution for 1 with a given convex function h. The K-ranks and K-sorts of $\mathbf{x}$ w.r.t $\boldsymbol{a}$ evaluated using ($\boldsymbol{b}$, $\boldsymbol{y}$) are respectively:*

$$\widetilde{R}(\mathbf{a}, \mathbf{x}; \mathbf{b}, \mathbf{y}) := n\mathbf{a}^{-1} \circ \left(P_\star \overline{\mathbf{b}}\right) \in [0, n]^n$$
$$\widetilde{S}(\mathbf{a}, \mathbf{x}; \mathbf{b}, \mathbf{y}) := \mathbf{b}^{-1} \circ \left(P_\star^T \mathbf{x}\right) \in \mathbb{O}_m$$

$\tilde{R} \xrightarrow{outputs}$ vector of size n containing a continuous rank $\forall\ x_i$ which can be seen as n times an empirical CDF value in [0, 1] view as a convex mixture of the CDF values b_j of the y_j onto which each x_i is transported. $\tilde{S} \xrightarrow{outputs}$ split-quantile operator outputting m increasing barycenters of some of the entries in x.

Computations and Non-differentiability. The previous operators have very small practical applications due to the complexity of $O(nm(n+m))$ to solve an OT problem far most costly than usual sorting and the non differentiable aspect of those operators. And worse the Jacobian $\frac{\partial P_*}{\partial x}$ is alike R, null almost everywhere. That is where regularized OT comes into minds.
$\tilde{R}$ and $\tilde{S}$ are expressed using the optimal solution P_* to the linear program in Eq.1. But we saw that P_* is not differentiable w.r.t inputs , $\mathbf{x}$ nor parameters $\mathbf{b}$, $\mathbf{y}$. Instead, we can use a differentiable alternative to OT using entropic regularization. The optimal regularized transport plan is a dense matrix, that ensures differentiability everywhere w.r.t. both a and x.

1.4 Smoothing of Kantorovich rank and sort operators with entropic regularization of OT and Sinkhorn algorithm

Let's consider the following regularized version of the OT problem with strength $\epsilon > 0$:

$$P_\star^\varepsilon := \underset{P \in U(\mathbf{a},\mathbf{b})}{\operatorname{argmin}} \langle P, C_{\mathbf{xy}} \rangle - \varepsilon H(P) \quad , \text{ where } \quad H(P) = -\sum_{i,j} P_{ij} \left(\log P_{ij} - 1 \right) \tag{3}$$

It can be shown that $P_\star^\varepsilon$ has the factorized form $\mathbf{D}(\mathbf{u}) K \mathbf{D}(\mathbf{v})$, with $K = \exp\left(-C_{\mathbf{xy}}/\varepsilon\right)$, and $\mathbf{u} \in \mathbf{R}^n$ and $\mathbf{v} \in \mathbf{R}^m$ are fixed points of the Sinkhorn algorithm defined in Alg.1.

Definition 2 *(Sinkhorn Rank and Sort). Given a regularization strength $\epsilon > 0$, run Alg.1 to define:*

$$\widetilde{R}_\varepsilon(\mathbf{a}, \mathbf{x}; \mathbf{b}, \mathbf{y}) := n \mathbf{a}^{-1} \circ \mathbf{u} \circ K(\mathbf{v} \circ \overline{\mathbf{b}}) \in [0, n]^n$$
$$\widetilde{S}_\varepsilon(\mathbf{a}, \mathbf{x}; \mathbf{b}, \mathbf{y}) := \mathbf{b}^{-1} \circ \mathbf{v} \circ K^T(\mathbf{u} \circ \mathbf{x}) \in \mathbb{R}^m$$

Algorithm 1 Sinkhorn

Inputs: $\mathbf{a}, \mathbf{b}, \mathbf{x}, \mathbf{y}, \epsilon, h, \eta$
$C_{\mathbf{xy}} \leftarrow [h(y_j - x_i)]_{i,j}$
$K \leftarrow e^{-C_{\mathbf{xy}}/\epsilon}$, $\mathbf{u} = \mathbf{1}_n$;
while $\Delta(\mathbf{v} \circ K^T\mathbf{u}, \mathbf{b}) < \eta$ **do**
  $\mathbf{v} \leftarrow \mathbf{v}/K^T\mathbf{u}, \mathbf{u} \leftarrow \mathbf{a}/K^T\mathbf{v}$
end while
return $\mathbf{u}$, $\mathbf{v}$, K

Sensitivity to ϵ

The bigger ϵ, the closer $P_\star^\varepsilon$ to matrix $\mathbf{a}\mathbf{b}^T$, and therefore all entries of $\widetilde{R}_\varepsilon$ collapse to the average of $n\overline{\mathbf{b}}$, while all entries of $\widetilde{S}_\varepsilon$ collapse to the weigted average (using $\mathbf{a}$) of $\mathbf{x}$. Although choosing a small value for ϵ might seem natural, in the sense that $\widetilde{R}_\varepsilon$, $\widetilde{S}_\varepsilon$ approximate more faithfully R, S, one should not forget that this would result in recovering the deficiencies of R, S in terms of differentiability. When learning with such operators, it may therefore be desirable to use a value for ϵ that is large enough to ensure $\partial P_\epsilon / \partial x$ has non-null entries. ϵ is often set to 10^{-2} or 10^{-3} when $\mathbf{x}$, $\mathbf{y}$ lie in $[0, 1]$. ϵ is kept fixed throughout Alg. 1, but speedups can be performed using scheduling.

Parallelization

The computation of K and K^T applied to $\mathbf{u}$ and $\mathbf{v}$ can be parallelized to accelarate the computation. See paper for further details.

Numerical Stability

When using small values of ϵ, the paper recommends to switch to the log domain with the following modfied updates:

$$\begin{aligned} \boldsymbol{\alpha} &\leftarrow \varepsilon \log \mathbf{a} + \min_c \left(C_{\boldsymbol{x},\mathbf{y}} - \boldsymbol{\alpha} \mathbf{1}_m^T - \mathbf{1}_n \boldsymbol{\beta}^T \right) + \boldsymbol{\alpha} \\ \boldsymbol{\beta} &\leftarrow \varepsilon \log \mathbf{b} + \min_c \left(C_{\boldsymbol{x_n},\mathbf{y}}^T - \mathbf{1}_m \boldsymbol{\alpha}^T - \boldsymbol{\beta} \mathbf{1}_n^T \right) + \boldsymbol{\beta} \end{aligned} \tag{4}$$

In this modified version, the min and its arguments corresponds to the soft minimum operator which improves the stability of exponential suming.

Choice of loss

Any convex function h can be used. Common choices are $h(u) = \|u\|^p$ with $p = 1, 2$.

Another important result form OT states that P_ϵ doesn't change under application of an increasing map to either $\mathbf{x}$ or $\mathbf{y}$. This property can be used to improve the stability of Alg. 1 by:

1. Setting y in the grid $[0, 1]$
2. Applying the squashing map:

$$\tilde{g} : \mathbf{x} \mapsto g \left(\frac{\mathbf{x} - \left(\mathbf{x}^T \mathbf{1}_n\right) \mathbf{1}_n}{\frac{1}{\sqrt{n}} \left\| \mathbf{x} - \left(\mathbf{x}^T \mathbf{1}_n\right) \mathbf{1}_n \right\|_2} \right) \tag{5}$$

to entry $\mathbf{x}$.

Summarizing the choices above, we obtain the modified Sinkhorn Alg. 2.

Algorithm 2 Sinkhorn Ranks/Sorts

Inputs: $(\mathbf{a}_s, \mathbf{x}_s)_s \in (\Sigma_n \times \mathbb{R}^n)^S, (\mathbf{b}, \mathbf{y}) \in \Sigma_m \times \mathbb{O}_m, h, \varepsilon, \eta, \widetilde{g}$
$\forall s, \overline{\mathbf{x}}_s = \bar{g}(\mathbf{x}_s), C_s = [h(y_j - (\overline{\mathbf{x}}_s)_i)]_{ij}, \boldsymbol{\alpha}_s = \mathbf{0}_n, \boldsymbol{\beta}_s = \mathbf{0}_m$
while $\max_s \Delta\left(\exp\left(C^{T'}_{\mathbf{x}_s} - \mathbf{1}_m \boldsymbol{\alpha}_s^T - \boldsymbol{\beta}_s \mathbf{1}_n^T\right)\mathbf{1}_n, \mathbf{b}\right) < \eta$; **do**
$\forall s, \boldsymbol{\beta}_s \leftarrow \varepsilon \log \mathbf{b}_s + \min_c\left(C_s^T - \mathbf{1}_m \boldsymbol{\alpha}_s^T - \boldsymbol{\beta}_s \mathbf{1}_n^T\right) + \boldsymbol{\beta}_s$
$\forall s, \boldsymbol{\alpha}_s \leftarrow \varepsilon \log \mathbf{a}_s + \min_e\left(C_s - \boldsymbol{\alpha}_s \mathbf{1}_m^T - \mathbf{1}_n \boldsymbol{\beta}_s^T\right) + \boldsymbol{\alpha}_s$
end while
$\forall s, \widetilde{R}_e(\mathbf{x}_s) \leftarrow \mathbf{a}_s^{-1} \circ \exp\left(C_{\mathbf{x}_\| \mathbf{y}} - \boldsymbol{\alpha}_s \mathbf{1}_m^T - \mathbf{1}_n \boldsymbol{\beta}_s^T\right)\overline{\mathbf{b}}$
$\forall s, \widetilde{S}_e(\mathbf{x}_s) \leftarrow \mathbf{b}_s^{-1} \circ \exp\left(C^T_{\mathbf{x}_s \mathbf{y}} - \mathbf{1}_m \boldsymbol{\alpha}_s^T - \boldsymbol{\beta}_s \mathbf{1}_n^T\right)\mathbf{x}_s$
return $\left(\widetilde{R}_\varepsilon(\mathbf{x}_s), \widetilde{S}_\varepsilon(\mathbf{x}_s)\right)_s$

1.5 Use case: learning with Smoothed Ranks and Sorts

Differentiable approximation of the top-k loss

Top-k loss Given $1, ..., L$ labels and Ω a set of points, a parametrized multiclass classifier on Ω is a function $f_\theta : \Omega \to \mathbb{R}^L$. The function selects the class attributed to ω by taking $l^\star = \text{argmax}_l [f_\theta(\omega)]_l$.

To train this model, a classical way is to minimize the cross entropy which results in solving

$$\min_\theta \ \Sigma_i \ \mathbf{1}_L^T log f_\theta(\omega_i) - [f_\theta(\omega_i)]_{l_i}$$

We want to obtain a differentiable version of the $0/1$ loss. if $l^\star \in argmax_l [f_\theta(\omega)]_l$ reduces to stating that the entry at $l^\star$ of the vector $R(f_\theta(\omega))$ is at position L.

Given a labelled pair (ω, l), the $0/1$ loss is therefore:

$$\mathcal{L}_{0/1}(f_\theta(\omega), l) = H\left(L - [R(f_\theta(\omega)]_l\right) \tag{6}$$

Where H is the Heavyside function: $H(u) = \mathbf{1}_{\{u<0\}}$.

The k-top expression proposed in 6 is not differentiable as H and R are discontinuous. The following smoothing of 6 is therefore proposed:

$$\widetilde{\mathcal{L}}_{k,\varepsilon}(f_\theta(\omega), l) = J_k\left(L - \left[\widetilde{R}_\varepsilon\left(\frac{\mathbf{1}_L}{L}, f_\theta(\omega); \frac{\mathbf{1}_L}{L}, \frac{\overline{\mathbf{1}}_L}{L}, h\right)\right]_l\right) \tag{7}$$

With J_k an increasing activation function, in the paper the ReLu function $J_k(u) = max(0, u-k-1)$ is considered.

2 Critical Analysis of the paper

This paper, using Optimal Transort, defines the Kantorovitch operators $\tilde{R}$ and $\tilde{S}$. Using entropic regularization and both the Sinkhorn algorithm and a modified version, it defines a new methodology to smooth these operators. Sorting is used in many methods of machine learning and the smoothed version of the sorting operator makes the task of optimizing discontinuous functions involving sort procedures possible. Regarding the usecases on which the methodology was applied: for the CNN for MNIST recognition, the method as performed better than state-of-the-art for the sorting task [GWZE19].

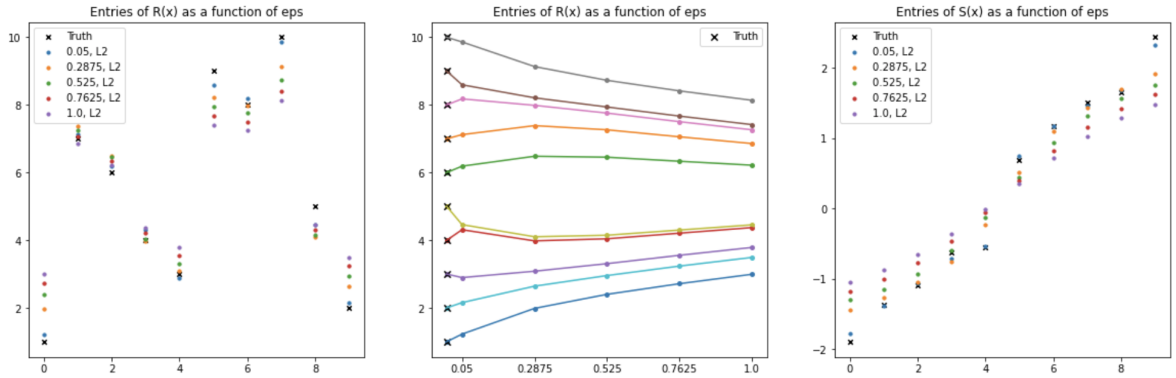
3 Implementation and Experiments

You can find our implementation of the paper, using python and numpy library in here, there are also an implementation of that algorithm, provided by google, implemented using JAX and TensorFlow. This allows them to conduct experiments with differention and optimisation of loss function, that include sorting procedure, since the JAX and TensorFlow include the auto-differentiation packages. In our implementation however, we have not repeated the experiments concerning the trainning of neural architectures, but implemented the soft sorting technique itself.

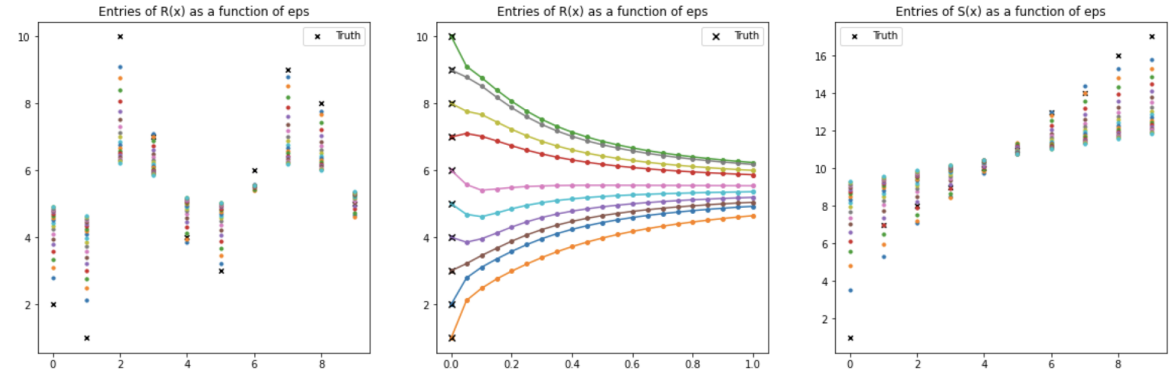
We have conducted the experiments to check influence of the parameter of regularisation ϵ, introduced in

$$P_\star^\varepsilon := \underset{P \in U(\mathbf{a},\mathbf{b})}{\operatorname{argmin}} \langle P, C_{\mathbf{xy}} \rangle - \varepsilon H(P) \quad , \text{ where } \quad H(P) = -\sum_{i,j} P_{ij} \left(\log P_{ij} - 1 \right)$$

Article proposes two two types of soft sort realisation. One, which is very straight forward, but struggles with low values of epsilon, due to explosions in exponential term. This have allowed to conduct experiments with epsilon values larger then 0.01.



In the experiments, we have observed, that with big values of epsilon the result converges very fast and keeps the ordering correct, but the scale of values are far from actual ranking. This also corresponds with the paper. To further extend the analysis, we have implemented the numerically stable version, which operates in the logarithm domain. However, the exponential term is still present in calculations, which by our observation makes the numerically stable algorithm behave unexpectedly with values of epsilon smaller then 1e-5.



Even though it may look like, its optimal to take value of epsilon as small as possible, to approximate the exact solution. We noted, that with lover values of epsilon, the Sinkhorn approximation takes longer to converge.

4 Conclusion

In the end, the link between sorting and OT in 1D lead us to those proxies of rank and sort. Using Sinkhorn iterations allow us to have automatically differentiated steps for sorting using OT. Numerical stability allow us to introduce these operators in various settings (e.g: smooth extensions of test-time metrics using sorting), then used as training losses. In the article, Sinkhorn sorting gives a smooth approximation of quantiles for least-quantile regression problems and an alternative to cross-entropy that may replace the 0/1 loss in multiclass classification. The differentiable aspect of this method allows for gradient dynamics reminiscent of rank based dynamics. Those dynamics are defined by the effort (gradient) produced by each player function of their rank. Sinkhorn is then interpreted as a smooth mechanism for such dynamics. However, choosing a cost function h, target vector y and

squashing function g are not supposed to influence the vector of Sinkhorn ranks or sorted values when $\epsilon \to 0$ (we always converge towards R,S).

Those assumptions constraint the differentiability of $\tilde{R}_\epsilon$ and $\tilde{S}_\epsilon$ when $\epsilon \geq 0$. The article claims that for numerical stability and consistent gradients through iterations, scaling inputs in [0, 1] is essential.

References

[GWZE19] Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. Stochastic optimization of sorting networks via continuous relaxations, 2019.