TP - SLICED WASSERSTEIN FLOW AND COLOR TRANSFER

ANATOLE GALLOUËT

contact: gallouet(at)math.univ-lyon1.fr

In this work, we implement an estimator of the Sliced Wasserstein distance between two measures. We then compute its gradient with respect to the position of one of the two measures in order to perform a gradient descent on this distance. This gradient descent can be implemented on measures representing RGB images, allowing us to apply the color palette of one image to another, a process known as color transfer.

Goal of this work. This practical work aims to:

- Discover the sliced variant of optimal transport.
- Implement Sliced Wasserstein flows on a simple problem.
- Study gradient descent parameters in a practical setting.
- Learn how to use basic PyTorch tools.
- Observe the speed difference between CPU and GPU computations.

The Sliced Wasserstein distance. The main idea behind Sliced Wasserstein (SW) is that there exists a closed form formula for the Wasserstein distance in dimension 1. It reads for $\mu, \nu \in \mathcal{P}(\mathbb{R})$,

$$W_p^p(\mu,\nu) = \int_0^1 |F_\mu^{-1}(t) - F_\nu^{-1}(t)|^p \mathrm{d}t$$

where F_μ and F_ν are the cumulative distribution functions of μ and ν. This formula becomes even more interesting when μ and ν are the sum of m dirac masses with uniform weights:

$$\mu = \frac{1}{m}\sum_{i=1}^m \delta_{x_i} \text{ and } \nu = \frac{1}{m}\sum_{i=1}^m \delta_{y_i}. \tag{1}$$

It can then be expressed as

$$W_p^p(\mu,\nu) = \frac{1}{m}\sum_{i=1}^m \left\|x_{\sigma(i)} - y_{\kappa(i)}\right\|^p,$$

where $\left(x_{\sigma(i)}\right)_{1\le i\le m}$ and $\left(y_{\kappa(i)}\right)_{1\le j\le m}$ are the sorted versions of $(x_i)_{1\le i\le m}$ and $(y_i)_{1\le i\le m}$. The computational benefits of this expression are significant, as the computational cost of the 1D Wasserstein distance is dominated by sorting both point clouds, which is $\mathcal{O}(m\log(m))$. The SW distance is a generalization of this idea to higher dimension, taking the 1D Wasserstein of projected measures on a direction θ and integrating over all possible directions $\theta \in \mathcal{S}^{d-1}$. Let $\mu, \nu \in \mathcal{P}(\mathbb{R}^d)$, the Sliced Wasserstein distance is

$$\mathrm{SW}_p^p(\mu,\nu) = \int_{\mathcal{S}^{d-1}} \mathrm{W}_p^p(P_{\theta\#}\mu, P_{\theta\#}\nu)\mathrm{d}\theta$$

where $P_\theta : \mathbb{R}^d \to \mathbb{R}$ is the projection in the direction θ, and $P_{\theta\#}\mu$ is the pushforward of μ by P_θ defined for $A \subset \mathbb{R}$ by $P_{\theta\#}\mu(A) = \mu(P_\theta^{-1}(A))$. Since $P_{\theta\#}\mu$ (resp. ν) are 1-d measures, we can compute explicitly the distance $\mathrm{W}_p^p(P_{\theta\#}\mu, P_{\theta\#}\nu)$ using the above formula. When both measures are the sum of m uniform Dirac masses as in (1), SW can be written

$$\mathrm{SW}_2^2(\mu,\nu) = \int_{\mathcal{S}^{d-1}} \frac{1}{m}\sum_{i=1}^m \langle x_{\sigma_\theta(i)} - y_{\kappa_\theta(i)}|\theta\rangle^2 \mathrm{d}\theta$$

where $\langle\cdot|\cdot\rangle$ is the scalar product in $\mathbb{R}^d$ and σ_θ and κ_θ are permutations such that $(\langle x_{\sigma_\theta(i)}|\theta\rangle)_i$ and $(\langle y_{\kappa_\theta(i)}|\theta\rangle)_i$ are sorted.

Sliced Wasserstein flows. In this part, we are interested in continuously moving a point cloud toward another by doing a gradient descent of the Sliced Wasserstein distance. Consider the Sliced Wasserstein distance between two discrete measures $\mu = \frac{1}{m}\sum_{i=1}^m \delta_{x_i}$ and $\nu = \frac{1}{m}\sum_{i=1}^m \delta_{y_i}$ as a function of $x \in (\mathbb{R}^d)^m$, where ν is a fixed measure, namely a function $\mathcal{E}_\nu : (\mathbb{R}^d)^m \to \mathbb{R}$ defined by

$$\mathcal{E}_\nu(x) = \mathrm{SW}_2^2(\mu,\nu) = \int_{\mathcal{S}^{d-1}} \frac{1}{m}\sum_{i=1}^m \langle x_{\sigma_\theta(i)} - y_{\kappa_\theta(i)}|\theta\rangle^2 \mathrm{d}\theta \tag{2}$$

Then the i-th coordinates of the gradient of $\mathcal{E}$, given by

$$\nabla\mathcal{E}_\nu(x)_i = 2\int_{\mathcal{S}^{d-1}} \frac{1}{m}\langle x_i - y_{\kappa_\theta\circ\sigma_\theta^{-1}(i)}|\theta\rangle\theta \mathrm{d}\theta$$

which can be approximated by a Monte Carlo method with samples $(\theta_\ell)_{1\le\ell\le L}$ giving

$$\nabla\mathcal{E}_\nu(x)_i \approx \frac{2}{m}\sum_{\ell=1}^L \frac{1}{L}\langle x_i - y_{\kappa_{\theta_\ell}\circ\sigma_{\theta_\ell}^{-1}(i)}|\theta_\ell\rangle\theta_\ell$$

This defines a stochastic gradient descent of the form $x_{k+1} = x_k - \tau_k\nabla\mathcal{E}_\nu(x)$. For an appropriate choice of the learning rate sequence (τ_k), the iterates x_k converge toward y.

Color transfer. An image of m pixels can be represented as a vector $x \in (\mathbb{R}^3)^m$ where x_i is the three RGB color of the i-th pixel. To this image x can be associated a color palette which is a discrete measure

$$\mu_x = \frac{1}{m}\sum_{i=1}^m \delta_{x_i}.$$

Note that in this representation, we lose the ordering of the image pixels, and just focus on the colors it contains. Take another image $y \in (\mathbb{R}^3)^m$, and apply the previous gradient descent to μ_x such that it matches μ_y. Then if you watch the image x with its original pixels ordering but shifted in the space of colors you get your original image with the color palette of the second one.

FIGURE 1. An example of color transfer. Source (left), Color target (middle) and Transfered (right) images.

Expected work. The practical work to do is:

- Implement an estimator of the Sliced Wasserstein distance.
- Implement the computation of the gradient of the distance.
- Implement a SW gradient descent for color transfer and reproduce Figure 1

It is preferable to implement all computations using pytorch, and if possible with the data on GPU to improve computational speed. The source and color palette image are in the "./images" folder, there are three different sizes (small, medium and big), in order to test the validity of the code on small images, and then improving the performance of the algorithm to make it work on larger images. Images "source_big.jpg" and "colors_big.jpg" are of order $m = 10^6$ pixels, which means that without an efficient GPU implementation, convergence of the SW flow can be very slow.

Evaluation. You will be evaluated on your code (a .py file which should be executable from command line), and a small report explaining your choices of implementation, parameters and stopping criterion (preferably latex, html or markdown).

Interesting extensions. The gradient descent can be replaced by a Newton algorithm, by computing the Hessian of the function $\mathcal{E}$ defined in Equation (2). This can also give informations of the ideal learning rate for SW flows, see [2] for details. One can also define the SW barycenter between measures. For images, this implies that one can harmonize the color palette of several images. See [1] for details.

REFERENCES

[1] N. Bonneel, J. Rabin, G. Peyré, and H. Pfister, "Sliced and Radon Wasserstein barycenters of measures," *Journal of Mathematical Imaging and Vision*, vol. 51, no. 1, pp. 22–45, 2015.

[2] C. Vauthier, A. Korba, and Q. Mérigot, "Towards Understanding Gradient Dynamics of the Sliced-Wasserstein Distance via Critical Point Analysis," *arXiv preprint arXiv:2502.06525*, 2025.