Optimal transport theory

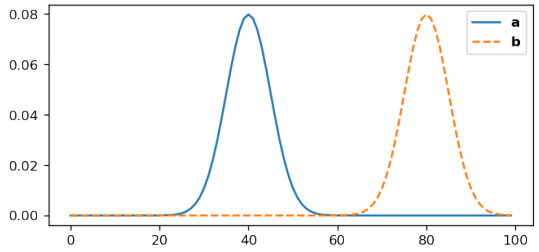
Student : Toby van Gastelen
Supervisors : Dr. Saad Yalouz - Pr. Dr. Lucas Visscher
Vrije Universiteit Amsterdam

I. INTRODUCTION

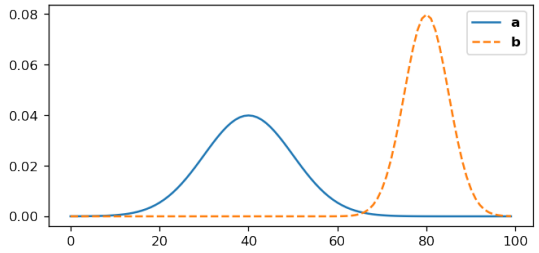
When we want to compare two vectors of the same length, e.g. $\mathbf{a}$ and $\mathbf{b}$, we usually resort to the l^2-norm of the difference:

$$\Delta = ||\mathbf{a} - \mathbf{b}||_2, \tag{1}$$

also known as the Euclidian distance. Larger Δ would in this case imply a greater dissimilarity between the vectors. We will now demonstrate the problem with this approach. Let us first look at two normalized Gaussians with the same standard deviation σ, but different means μ and look at the l^2-norm of the difference between the vectors containing the function evaluations for $\mathbf{x} = (0, 1, ..., 99)^T$, $a(x_i) = a_i$ and $b(x_i) = b_i$ (FIG. 1a). We properly normalize each of the vectors to represent a discrete probability distribution on $[0, 99]$. Let us also do the same thing, after increasing the σ of $\mathbf{a}$ from 5 to 10 (FIG. 1b). According to the l^2-norms of both examples, we find that the Gaussians with different σ (example two) are more similar as compared to the Gaussians with the same σ (example one). This is caused by the overlap between $\mathbf{a}$ and $\mathbf{b}$, due to the increased σ of $\mathbf{b}$. Let us now take a look what would happen if we calculate the Wasserstein distance W_2 for both examples (FIG. 1). Again, a larger value would imply a greater dissimilarity between the vectors. For this we used the **Python optimal transport**[1] (POT) library . Now we find that the difference between the two vectors is actually smaller in the first example, as compared to the second example, due to the similar shapes of the evaluated functions. We also know that when comparing two vectors containing the same shapes at different locations, W_2 equals the absolute distance between both of the centers of mass. Looking at example one this seems to be the case. Another nice property of the Wasserstein distance is that it allows us unambigouisly compare vectors of different dimension. The procedure can easily be extended to matrices and higher dimensional objects.



(a) Example one: Two vectors $\mathbf{a}$ and $\mathbf{b}$ containing the Gaussian function evaluations for $\mu = 40$ and $\sigma = 5$, and $\mu = 80$ and $\sigma = 5$, respectively. In this case we have $\Delta = 0.3359$ and $W_2 = 40.00$.



(b) Example two: Two vectors $\mathbf{a}$ and $\mathbf{b}$ containing the Gaussian function evaluations for $\mu = 40$ and $\sigma = 10$, and $\mu = 80$ and $\sigma = 5$, respectively. In this case we have $\Delta = 0.2907$ and $W_2 = 40.31$.

Figure 1: Comparison between two Gaussians using Δ and W_2.

II. WASSERSTEIN DISTANCE

Let us now dive a little bit deeper into this metric using mostly the theory presented in the book **Computational optimal transport**[2]. We first define the Wasserstein distance as

$$W_p(\mathbf{a}, \mathbf{b}) = \{\min_{\boldsymbol{\gamma}} \sum_{ij} \gamma_{ij} M_{ij}(p)\}^{1/p}, \tag{2}$$

for which $a(\mathbf{x}_i) = a_i$ and $b(\mathbf{y}_i) = b_i$. W_p can be thought of as a measure to compare the resemblance of two shapes. Here we define the cost matrix $\mathbf{M}$ as

$$M_{ij,p} = ||\mathbf{x}_i - \mathbf{y}_j||_2^p \rightarrow \mathbf{M}_p. \tag{3}$$

This matrix encodes the penalty of moving mass from $\mathbf{x}_i$ to $\mathbf{y}_j$. In the case of matrices $a(\mathbf{x}_i)$ and $b(\mathbf{y}_i)$ map the positions $\mathbf{x}_i/\mathbf{y}_i = (row, column)$ to the value of the matrix entry, i.e. $a([x_{i1}, x_{i2}]) = a_i$ and $b([y_{i1}, y_{i2}]) = b_i$. For $\mathbf{M}$ and $\boldsymbol{\gamma}$ we have

$$\dim(\mathbf{M}) = \dim(\boldsymbol{\gamma}) = \dim(\mathbf{a}) \times \dim(\mathbf{b}), \tag{4}$$

i.e. $\mathbf{M}_{\dim(\mathbf{a})\times\dim(\mathbf{b})}$ and $\boldsymbol{\gamma}_{\dim(\mathbf{a})\times\dim(\mathbf{b})}$. For this procedure to work both $\mathbf{a}$ and $\mathbf{b}$ need to abide by

$$\begin{aligned} \sum_i a_i &= 1, \\ \sum_i b_i &= 1 \\ a_i, b_i &\geq 0 \ \forall i, \end{aligned} \tag{5}$$

such that $\mathbf{a}$ and $\mathbf{b}$ can be thought of as probability densities. The following constraints are put on $\boldsymbol{\gamma}$:

$$\begin{aligned} \sum_{ij} \gamma_{ij} &= 1, \\ \gamma_{ij} &\geq 0, \forall i, j, \\ \sum_{i} \gamma_{ij} &= b_j, \\ \sum_{j} \gamma_{ij} &= a_i, \end{aligned} \tag{6}$$

which we will refer to as $\boldsymbol{\gamma} \in \mu(\mathbf{a}, \mathbf{b})$. Now $\boldsymbol{\gamma}$ represents a probability density such that

$$P(\mathbf{x}_i, \mathbf{y}_j) = \gamma_{ij} \geq 0. \tag{7}$$

In reality $\boldsymbol{\gamma}$ encodes the permutations that need to be carried out to transform one vector into the other, which is then weighted by $\mathbf{M}_p$. Regarding the Wasserstein distance the following relations hold:

$$\begin{aligned} &W_p(\mathbf{a}, \mathbf{b}) = 0 \text{ iff } \mathbf{a} = \mathbf{b}, \\ &W_p(\mathbf{a}, \mathbf{b}) = W_p(\mathbf{b}, \mathbf{a}), \\ &W_p(\mathbf{a}, \mathbf{b}) \leq W_p(\mathbf{a}, \mathbf{c}) + W_p(\mathbf{c}, \mathbf{b}), \forall \mathbf{a}, \mathbf{b}, \mathbf{c} \end{aligned} \tag{8}$$

for $p > 1$, which makes that W_p classifies as a distance in the space of probability densities. In determining W_p the objective is to find the minimizer $\boldsymbol{\gamma}$. For this we apply numerical optimization techniques. Algorithms to obtain W_p scale at best to order $O(n^2)$, where n is the dimension of the input vectors $\mathbf{a}$ and $\mathbf{b}$. To make numerical optimization more feasible for large input data we introduce the addition of an entropic term $\Omega(\boldsymbol{\gamma})$ weighted by constant $\lambda > 0$:

$$\begin{aligned} W_p^{\lambda}(\mathbf{a}, \mathbf{b}) &= \{\min_{\boldsymbol{\gamma}} \sum_{ij} (\gamma_{ij} M_{ij,p}) + \lambda \sum_{ij} \gamma_{ij} (\log(\gamma_{ij}) - 1)\}^{1/p} \\ &= \{\min_{\boldsymbol{\gamma}} \sum_{ij} (\gamma_{ij} M_{ij,p}) + \lambda \Omega(\boldsymbol{\gamma})\}^{1/p}, \end{aligned} \tag{9}$$

resulting in

$$\boldsymbol{\gamma}^*(\lambda) = arg \min_{\boldsymbol{\gamma}} \{\sum_{ij} (\gamma_{ij} M_{p,ij}) + \lambda \Omega(\boldsymbol{\gamma})\}. \tag{10}$$

This can in turn be optimized using the Sinkhorn algorithm. This algorithm scales to $O(n/\lambda^2)$. The algorithm is treated in more detail in SEC. III. For $p > 1$ the solution of $\boldsymbol{\gamma}$ (or approximate solution $\boldsymbol{\gamma}^*$) is indepent of chosen p, i.e. the solution is always the same for $p > 1$. This is why there is no need to write the exponent anymore in EQ. 10. Increasing the parameter λ results in faster convergence to the solution but also results in a larger error, as compared to $\lambda = 0$. After numerically obtaining the solution for $\boldsymbol{\gamma}$ (or $\boldsymbol{\gamma}^*$) we obtain W_p (or W_p^{λ}) by doing

$$W_p = (\sum_{ij} \gamma_{ij} M_{p,ij})^{1/p} = \mathrm{Tr}(\mathbf{M}_p^T \boldsymbol{\gamma})^{1/p}, \tag{11}$$

omitting the entropic part.

III. SIKNHORN ALGORITHM

Next, we will look at the Sinkhorn algorithm used to solve the minimization problem described in EQ. 10. To solve this problem we define the Lagrangian as

$$\mathcal{L}(\boldsymbol{\gamma}, \mathbf{v}, \mathbf{u}) = \sum_{ij} [\gamma_{ij} M_{ij,p} + \lambda \gamma_{ij} (\log(\gamma_{ij}) - 1)] + \begin{cases} -\sum_i v_i [\sum_j (\gamma_{ij}) - a_i] - \sum_j u_j [\sum_i (\gamma_{ij}) - b_j] & \text{if } \boldsymbol{\gamma} \in \mu(\mathbf{a}, \mathbf{b}) \\ \infty & \text{if } \boldsymbol{\gamma} \notin \mu(\mathbf{a}, \mathbf{b}) \end{cases} \tag{12}$$

where $\mathbf{v}$ and $\mathbf{u}$ contain the Lagrange multipliers that ensure that the constraints in EQ. 6 are satisfied.[3] The $+\infty$ condition is added to ensure $\mathcal{L}(\boldsymbol{\gamma}, \mathbf{v}, \mathbf{u})$ remains strictly convex in γ_{ij} for $\gamma_{ij} \in \mathbb{R}$. To remind ourselves: a function $f(x)$ is strictly convex[4] for $x \in \mathbb{R}$ if

$$\forall x_1 \neq x_2 \in \mathbb{R}, \forall t \in (0, 1) : f(t x_1 + (1 - t) x_2) < t f(x_1) + (1 - t) f(x_2). \tag{13}$$

Implications of this are that that

$$\frac{d^2 f(x)}{dx^2} > 0, \forall x \in \mathbb{R} \tag{14}$$

and $f(x)$ contains at most one minimum, which in turn is then the global minimum. We can then find this global minimum by solving $\frac{df(x)}{dx} = 0$, which has now only one solution. We can check that EQ. 14 holds for $\mathcal{L}(\boldsymbol{\gamma}, \mathbf{v}, \mathbf{u})$ by doing

$$\frac{\partial^2 \mathcal{L}(\boldsymbol{\gamma}, \mathbf{v}, \mathbf{u})}{\partial \gamma_{ij}^2} = \frac{\lambda}{\gamma_{ij}} > 0, \forall \gamma_{ij} \in \mu(\mathbf{a}, \mathbf{b}). \tag{15}$$

Now we can obtain $\boldsymbol{\gamma}^*$ by doing

$$\frac{\partial \mathcal{L}(\boldsymbol{\gamma}, \mathbf{v}, \mathbf{u})}{\partial \gamma_{ij}} = M_{ij,p} + \lambda \log \gamma_{ij} - v_i - u_j = 0. \tag{16}$$

The solution to this equation is

$$\gamma_{ij}^* = \exp(\frac{v_i}{\lambda}) \exp(\frac{u_j}{\lambda}) \exp(\frac{-M_{ij,p}}{\lambda}) = f_i g_j K_{ij}, \tag{17}$$

where the K_{ij} resembles the Gibbs kernel as used in statistical mechanics.[5] λ Can be thought of as a temperature parameter similar to $k_b T$. Making λ larger increasingly desensitizes the system to unoptimal permutations present in $\boldsymbol{\gamma}$. To make the notation more simple we briefly change to integral notation. We assume $\gamma^*(x, y)$ has the following form

$$\gamma^*(x, y) = f(x) g(y) K(x, y). \tag{18}$$

Now, we can rewrite the constraints as

$$\begin{aligned} \int \gamma^*(x, y) dy &= a(x), \\ \int \gamma^*(x, y) dx &= b(y). \end{aligned} \tag{19}$$

Note here that the first two constrains are now also satisfied, because of the constraints we placed on **a** and **b** (EQ. 5). Now we write

$$g(y) = \frac{b(y)}{\int f(x)K(x,y)dx}, \qquad f(x) = \frac{a(x)}{\int g(y)K(x,y)dy}, \tag{20}$$

which can be transformed into an iterative scheme

$$g^{n+1}(y) = \frac{b(y)}{\int f^{n}(x)K(x,y)dx}, \qquad f^{n+1}(x) = \frac{a(x)}{\int g^{n+1}(y)K(x,y)dy}. \tag{21}$$

When going back to the matrix/vector notation we can now write the complete algorithm, which is presented in the next section.

A. The algorithm

- Initialization:

$$\mathbf{g}^0 = (1, 1, ..., 1)^T, \text{ where } \dim(\mathbf{g}) = \dim(\mathbf{b})$$
$$\mathbf{f}^0 = \mathbf{a}^0$$
$$\gamma_{ij}^{*0} = f_i^0 K_{ij} g_j^0 \leftrightarrow \boldsymbol{\gamma}^{*0} = \text{diag}(\mathbf{f}^0)\mathbf{K}\text{diag}(\mathbf{g}^0)$$
$$\textbf{iterations} = 0$$

- While **iterations** < **max-iterations** and $||\mathbf{g}\odot(\mathbf{K}^T\mathbf{f})-\mathbf{b}||_2 >$ **threshold**:

$$g_i^{n+1} = \frac{b_i}{(\mathbf{K}^T\mathbf{f}^n)_i}$$
$$f_i^{n+1} = \frac{a_i}{(\mathbf{K}\mathbf{g}^{n+1})_i}$$
$$\gamma_{ij}^{*n+1} = f_i^{n+1} K_{ij} g_j^{n+1} \leftrightarrow \boldsymbol{\gamma}^{*n+1} = \text{diag}(\mathbf{f}^{n+1})\mathbf{K}\text{diag}(\mathbf{g}^{n+1})$$
$$\textbf{iterations} \rightarrow \textbf{iterations} + 1,$$

where $\odot$ represents the Hadamard product. One should be careful when changing the order of updating **g** and **f**, due to how the convergence condition is defined.

---

[1] Rémi Flamary and Nicolas Courty. Python optimal transport. `https://pot.readthedocs.io/en/stable/`, 2019. Accessed: 28-01-2020.

[2] Gabriel Peyré and Marco Cuturi. *Foundations and Trends in Machine Learning*, 11:355–607, 2019.

[3] Paul Dawkins. Paul's online notes, section 3-5: Lagrange multipliers, 2020. Accessed: 29-01-2020.

[4] Markus Grasmair. Basic properties of convex functions. Accessed: 29-01-2020.

[5] A. Bovier, F. Dunlop, A. van Enter, F. den Hollander, and J. Dalibard. *Mathematical statistical physics*. Elsevier, 2006.