ENS Lyon
2024/2025 **Mathurin Massias, Titouan Vayer, Quentin Bertrand**

Homework 1 : Basics of OT

You have three weeks to do this homework: it must be return by Wednesday, November 20.

- Send the homework to titouan.vayer@inria.fr, mathurin.massias@inria.fr and quentin.bertrand@inria.fr with the header "Homework 1 Name 1".
- For the maths send a scan by mail or give it by hand on 20th november.

- Exercise 1: Dual of OT. -

Let $\alpha = \sum_{i=1}^n a_i \delta_{\mathbf{x}_i}, \beta = \sum_{j=1}^m b_j \delta_{\mathbf{y}_j}$ be two discrete probability measures. Let $\mathbf{C} = (c(\mathbf{x}_i, \mathbf{y}_j))_{ij}$ be the cost matrix between the samples. Consider the OT problem

$$\min_{\mathbf{P} \in U(\mathbf{a}, \mathbf{b})} \langle \mathbf{P}, \mathbf{C} \rangle \, . \tag{Primal}$$

Using the KKT conditions, show that (Primal) admits the dual formulation

$$\max_{\substack{\mathbf{f} \in \mathbb{R}^n, \mathbf{g} \in \mathbb{R}^m \\ \forall (i,j) \in [n] \times [m], f_i + g_j \leq C_{i,j}}} \langle \mathbf{f}, \mathbf{a} \rangle + \langle \mathbf{g}, \mathbf{b} \rangle \, . \tag{Dual}$$

Deduce that

- if $\mathbf{P}^\star$ is a solution of (Primal) and $(\mathbf{f}^\star, \mathbf{g}^\star)$ is a solution of (Dual) then $\langle \mathbf{P}^\star, \mathbf{C} \rangle = \langle \mathbf{f}^\star, \mathbf{a} \rangle + \langle \mathbf{g}^\star, \mathbf{b} \rangle$
- for any $(i, j) \in \mathrm{supp}(\mathbf{P}^\star)$, $f_i^\star + g_j^\star = C_{i,j}$ where $\mathrm{supp}(\mathbf{P}^\star) = \{(i, j) : P_{ij}^\star > 0\}$.

- Exercise 2: basics of the POT library. -

This second exercise aims at getting started with the `POT` library `https://pythonot.github.io/` for optimal transport. We will use `jupyter notebook` for all practical sessions. Some important points to remember when working with `Python`:

```
import math                        # import a package
import numpy as np                 # import a package under an alias
from sklearn import linear_model   # import a submodule
from os import mkdir               # import a specific function
```

First install `POT` and the `scikit-learn` library. We will use the following data

```
from sklearn.datasets import make_blobs
offset = 0.5
seed = 42
centers = np.array([
    [0, 0],
    [offset, 0],
])
X, c = make_blobs(n_samples=50, centers=centers, n_features=2,
                  random_state=seed, cluster_std=0.05, shuffle=False)
X = X[c == 0] # source distribution
Y = X[c == 1] # target distribution
```

(i) Compute two cost matrices between the samples: the first one should be $(\|\mathbf{x}_i - \mathbf{y}_j\|_2)_{ij}$ and the second one $(\|\mathbf{x}_i - \mathbf{y}_j\|_2^2)_{ij}$ (you can use the `ot.dist` function).

(ii) Compute the weights of each point in the distributions, we can take uniform weights for simplicity.

(iii) For each cost matrix calculate the optimal transport plan between the source distribution and the target distribution. For this you can use the `ot.emd` function.

(iv) Plot the two optimal transport plans along with the source and target samples. You can use the following code.

```
def plot2D_samples_mat(xs, xt, G, ax, thr=1e-8, **kwargs):
    """Plot 2D samples and the corresponding coupling between them.

    Parameters
    ----------
    xs : np.array (n_samples_source, 2)
        The source points
    xt : np.array (n_samples_target, 2)
        The target points
    G : np.array (n_samples_source, n_samples_target)
        The coupling
    ax : matplotlib.axes._axes.Axes
        Axes object for the plot
    thr : float, optional
        Threshold parameter for showing an edge, by default 1e-8
    kwargs: dictionary
        Other arguments for the plot (color, alpha...)
    """
    if ('color' not in kwargs) and ('c' not in kwargs):
        kwargs['color'] = 'k'
    mx = G.max()
    if 'alpha' in kwargs:
        scale = kwargs['alpha']
        del kwargs['alpha']
    else:
        scale = 1
    for i in range(xs.shape[0]):
        for j in range(xt.shape[0]):
            if G[i, j] / mx > thr:
                ax.plot([xs[i, 0], xt[j, 0]],
                        [xs[i, 1], xt[j, 1]],
                        alpha=G[i, j] / mx * scale, **kwargs)
```

(v) What is the big difference between the two optimal plans ?

(vi) What is the value of the Wasserstein distance in these two cases ?