Semi-discrete optimal transport - sdot package

Jocelyn Meyron

July 19, 2019

Code walkthrough

We start by detailing the external dependencies this project relies on before detailing the provided examples.

External dependencies

They are all contained inside the `lib` directory.

The three following directories are three different implementations of an algorithm for computing Laguerre diagrams (restricted Voronoi diagrams).

- `laguerre`: see [1] for more details;
- `geogram`: utilizes the geogram library, see [2] for more details;
- `NNRVD`: utilizes part of the Revoropt library, see [3] for more details.

We then created Python bindings for the C++ functions using the pybind11 library (also included as an external dependency).

Code structure

The C++ functions for computing the Laguerre diagrams are the C++ files located inside the `src/sdot/` directory. For each implementation corresponds a backend. Each backend defines a `Density_*` class which will represent a source density in an optimal transport setting, see the `sdot.backend` module.

In each backend, we differentiate between 2D densities, densities supported on triangulated surfaces and densities supported on tetrahedrizations. If you want to use the `laguerre` backend for densities supported on triangulated surfaces, you would do the following:

```
from sdot.backend import LaguerreBackend
Density = LaguerreBackend().available_settings["Surface"]
```

The possible keys for available_settings are `Plane`; `Surface` and `3D`. The available settings are detailed in the next table.

Table 1: Available settings for each backend.

	`laguerre`	`geogram`	`NNRVD`
`Plane`	YES	YES	YES
`Surface`	YES	YES	YES
`3D`	NO	YES	NO

The optimal transport algorithms (damped Newton and BFGS) can be found in the `sdot.optimal_transport` module.

Loading and writing OFF files is done using the `sdot.core.inout` module.

Other utility functions based on CGAL data structures such as an AABB tree (`TriangleSoup` class in the `sdot.aabb_tree` module) or some functions to compute *conforming* centroids (`conforming_lloyd_2` and `conforming_lloyd_3` in the `sdot.cgal_utils` module).

Basics

- Creating a density:

```
from sdot.backend import NNRVDBackend
import sdot.core.inout as io
Density_2 = NNRVDBackend().available_settings["Surface"] # Density on a triangulation
X, T = io.read_off("examples/assets/sphere.off") # Mesh to define the density on
mu = Density_2(X, T) # If no values are provided, creates a uniform density
V = np.repeat(1.0, len(X)) # One value per vertex
mu = Density_2(X, T, V) # Creates a piecewise affine density
```

- Computing Laguerre cells areas and derivatives:

```
import numpy as np
Y = np.loadtxt("examples/assets/clouds/sphere_1k.cloud") # A point cloud
psi = np.zeros(len(Y)) # The weights of the Laguerre diagram
nu = np.repeat(1.0, len(Y)) # The uniform target density on Y
A, DA = mu.kantorovich(Y, nu, psi=psi) # Returns G and DG
# Use mu.res to access other results such as:
C = mu.res["C"] # The centroids of the Laguerre cells
T = mu.res["T"] # The combinatorics of the dual triangulation of the Laguerre diagram
```

Examples

The examples are located in the `examples` directory. There are two of them.

- `ot.py`: compute the optimal transport between a uniform measure supported on a triangulated surface and a uniform measure supported on a 3D point cloud

More details on the usage of the examples can be found in the beginning of the two files.

Example images, meshes and clouds are provided in the `examples/assets` directory.

References

[1] Q. Mérigot, J. Meyron, and B. Thibert, "An algorithm for optimal transport between a simplex soup and a point cloud," *SIAM Journal on Imaging Sciences*, vol. 11, no. 2, pp. 1363–1389, 2018.

[2] B. Lévy, "A numerical algorithm for L^2 semi-discrete optimal transport in 3D," *ESAIM: Mathematical Modelling and Numerical Analysis*, vol. 49, no. 6, pp. 1693–1715, 2015.

[3] V. Nivoliers, "Échantillonnage pour l'approximation de fonctions sur des maillages," PhD thesis, Université de Lorraine, 2012.