Journal of Machine Learning Research 22 (2021) 1-8 Submitted 5/20; Revised 1/21; Published 4/21

POT: Python Optimal Transport

Rémi Flamary REMI.FLAMARY@POLYTECHNIQUE.EDU
Nicolas Courty COURTY@UNIV-UBS.FR
Alexandre Gramfort ALEXANDRE.GRAMFORT@INRIA.FR
Mokhtar Z. Alaya MOKHTARZAHDI.ALAYA@GMAIL.COM
Aurélie Boisbunon ABB@MYDATAMODELS.COM
Stanislas Chambon SCHAMBON@THERAPIXEL.COM
Laetitia Chapel LAETITIA.CHAPEL@IRISA.FR
Adrien Corenflos ADRIEN.CORENFLOS@AALTO.FI
Kilian Fatras KILIAN.FATRAS@IRISA.FR
Nemo Fournier NEMO.FOURNIER@ENS-LYON.ORG
Léo Gautheron LEO.GAUTHERON@UNIV-ST-ETIENNE.FR
Nathalie T.H. Gayraud NATHALIETHG@FB.COM
Hicham Janati HICHAM.JANATI@INRIA.FR
Alain Rakotomamonjy A.RAKOTOMAMONJY@CRITEO.COM
Ievgen Redko IEVGEN.REDKO@UNIV-ST-ETIENNE.FR
Antoine Rolet ANTOINE.ROLET@IIP.IST.I.KYOTO-U.AC.JP
Antony Schutz ANTONY.SCHUTZ@UNICE.FR
Vivien Seguy VIVIENSEGUY@GMAIL.COM
Danica J. Sutherland DSUTH@CS.UBC.CA
Romain Tavenard ROMAIN.TAVENARD@UNIV-RENNES2.FR
Alexander Tong ALEXANDER.TONG@YALE.EDU
Titouan Vayer TITOUAN.VAYER@ENS-LYON.FR

Editor: Andreas Mueller

Abstract

Optimal transport has recently been reintroduced to the machine learning community thanks in part to novel efficient optimization procedures allowing for medium to large scale applications. We propose a Python toolbox that implements several key optimal transport ideas for the machine learning community. The toolbox contains implementations of a number of founding works of OT for machine learning such as Sinkhorn algorithm and Wasserstein barycenters, but also provides generic solvers that can be used for conducting novel fundamental research. This toolbox, named POT for Python Optimal Transport, is open source with an MIT license.

Keywords: Optimal transport, divergence, optimization, domain adaptation

1. Introduction and motivations

Optimal Transport (OT) is a field of mathematics which studies the geometry of probability spaces. Among its many contributions, OT provides a principled way to compare

and align probability distributions by taking into account the underlying geometry of the considered metric space. Examples where this approach is useful in machine learning (ML) are ubiquitous and include, for instance, prominent tasks such as training generative adversarial networks (GANs) (Arjovsky et al., 2017) where OT was successfully used to overcome the vanishing gradient problem, and domain adaptation (Courty et al., 2016) where its capacity to align two distributions was used to transfer a classifier across different domains. Other examples showing the versatility of OT are given by the averaging of population data in neuroscience (Gramfort et al., 2015), shape reconstruction (Bonneel et al., 2016) and learning word embeddings in natural language processing (Kusner et al., 2015). The success of OT in different scientific areas resides in two main features. On the one hand, OT-based Wasserstein distance compares favourably to popular f-divergences including popular Kullback-Leibler, Jensen-Shannon divergences and Total Variation distance, when the support of the two distributions is disjoint. On the other hand, and contrary to other popular integral probability metrics (IPMs) such as maximum mean discrepancy (MMD), the solution of the OT problem is given by a probabilistic (coupling) function that provides soft-assignments between the points in the supports of the two distributions, which can be used for their further matching. We refer the interested reader to the book of Villani (2008) for a complete theoretical treatment, and to the book of Peyré and Cuturi (2018) for a more comprehensive study on OT's computational aspects. With growing interest of the machine learning community in optimal transport, it is important to provide a simple and efficient framework with several OT solvers. We choose to conduct its development in the Python programming language, for its wide use in the machine learning community. Its high-level interactive nature makes it an appealing tool for both academic and industrial software developments, and several popular machine learning libraries (Pedregosa et al., 2011) and deep learning open source frameworks (Paszke et al., 2019; Abadi et al., 2015) are built upon it.

The Python Optimal Transport (POT) library takes advantage of Python to make Optimal Transport accessible to the machine learning community. It provides state-of-the-art algorithms to solve the regular OT optimization problems, and related problems such as entropic Wasserstein distance with Sinkhorn algorithm or barycenter computations. As POT is designed to be user focused and friendly, we have kept our toolbox easy to use and accessible with convention consistencies and syntax over all the available functions. Moreover, our toolbox depends only on standard open source libraries. Available under an MIT license, it is usable under many operating systems such as Linux, MacOSX or Windows. Furthermore, the generic solvers relying on for OT variants (group lasso regularization, Gromov Wasserstein) can also be used for solving novel OT formulations in fundamental research. Finally, we provide new users a quick start guide for POT in the documentation and several examples and notebooks for our different solvers. POT is greatly appreciated in the machine learning community, with more than 5K downloads each month on PyPI.

2. Existing Optimal Transport solvers

There exist several open-source optimal transport solvers, but most of them are companion codes of papers and are either outdated or not currently maintained. For instance,

we can mention the codes of Chizat (2016), which addresses unbalanced optimal transport, and of Farchi (2016), who proposes a Benamou-Brenier OT solver. From our perspective, the two most important current solvers are the OT Toolbox (OTT) (Schmitzer, 2018) and the OTJulia package (Zhang et al., 2020). The OT Toolbox proposes several OT solvers based on a C++ implementation with a Python binding. This toolbox has limited documentation, is rarely updated and is still considered a preliminary version. The OTJulia is a more complete package that can be seen as a Julia counterpart of POT. It implements several OT solvers with detailed documentation, but still misses some features related to recent algorithmic developments, and interestingly is a wrapper around POT for its exact EMD OT solver. Finally, GeomLoss (Feydy et al., 2019) is a more specific toolbox for solving very large scale Sinkhorn on CPU and GPU with a PyTorch API. It is based on the very elegant KeOps framework (Charlier et al., 2020), and is currently a better alternative to POT for using Sinkhorn losses in deep learning architectures.

	Package			
Features	POT	OTJulia	OTT	GeomLoss
EMD	✓	✓	✓	
EMD Variants (multiscale)			✓	
Sinkhorn	✓	✓	✓	✓
Stabilized Sinkhorn	✓	✓	✓	✓
Unbalanced/Partial OT	✓	✓	(✓)	
Regularized OT	✓	(✓)		
Wasserstein Barycenter	✓	✓	✓	(✓)
Gromov-Wasserstein	✓			
Sliced Wasserstein	✓			
Domain Adaptation	✓			
GPU implementation	(✓)			✓

Table 1: Feature comparisons between POT, OTJulia, OTT and GeomLoss.

3. Toolbox usage

The library provides recent state-of-the-art solvers for various optimal transport problems related to statistics and machine learning. As mentioned earlier, POT has sub-modules dedicated to different problems. In this section, we give a short description of the provided OT solvers.

Kantorovich optimal transport problems. This is the most typical OT problem. It seeks an optimal coupling $\boldsymbol{T}$ which minimizes the displacement cost of a discrete measure $\boldsymbol{a}$ to a discrete measure $\boldsymbol{b}$ with respect to a ground cost $\boldsymbol{M} \in \mathbb{R}^{n_1 \times n_2}$. In order to be a transport plan, $\boldsymbol{T}$ must be part of the set $\Pi(\mathbf{a}, \mathbf{b}) = \{\boldsymbol{T} \geq \mathbf{0}, \boldsymbol{T}\mathbf{1}_{n_2} = \boldsymbol{a}, \boldsymbol{T}^\top \mathbf{1}_{n_1} = \boldsymbol{b}\}$. When the ground cost is a metric, the optimal value of the OT problem is also a metric (Rubner et al., 2000; Cuturi and Avis, 2014) and is called the *Wasserstein distance*. In this discrete case, the OT problem is defined as

$$W_{\boldsymbol{M}}(\boldsymbol{a}, \boldsymbol{b}) = \min_{\boldsymbol{T} \in \Pi(\mathbf{a}, \mathbf{b})} \langle \boldsymbol{T}, \boldsymbol{M} \rangle \ , \tag{1}$$

which is a linear program. The optimization problem above is often adapted to include a regularization term for the transport plan $\boldsymbol{T}$, such as entropic regularization (Cuturi, 2013) or squared L2. For the entropic regularized OT problem, one may use the Sinkhorn Knopp algorithm (or variants), or stochastic optimization algorithms. POT has a simple syntax to solve these problems (see Sample 1).

Wasserstein barycenter problems. The notion of barycenters can be extended to the use of the Wasserstein distance, allowing one to define the Wasserstein barycenter for

Sample 1: OT matrix POT syntax

```
import ot
# a, b are 1D histograms (sum to 1 and positive)
# M is the ground cost matrix
T = ot.emd(a, b, M)  # exact linear program
T_reg = ot.sinkhorn(a, b, M, reg)  # entropic regularized OT
```

a set of N probability measures $\{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_N\}$ as follows:

$$\boldsymbol{a}^* = \arg\min_{\boldsymbol{a}} \sum_{i=1}^{N} \alpha_i W_{\boldsymbol{M}}(\boldsymbol{a}_i, \boldsymbol{a}), \quad \sum_{i=1}^{N} \alpha_i = 1. \tag{2}$$

When using entropic regularization, an efficient Bregman projection algorithm is provided in POT, on the other hand the unregularized problem is solved with LP solvers (see Sample 2).

Sample 2: OT barycenters POT syntax

```
# A is a n*d matrix containing d 1D histograms
# M is the ground cost matrix
ba = ot.barycenter(A, M, reg)  # entropic regularization
ba0 = ot.lp.barycenter(A, M)  # unregularized barycenter (slow)
```

Other solvers and tools. Different OT solvers are available in sub-modules of POT. **`ot.da`** contains methods and classes with a scikit-learn compatible API for domain adaptation applications. **`ot.gromov`** and **`ot.partial`** contain respectively solvers for Gromov-Wasserstein and partial OT problems. Finally we provide in **`ot.optim`** several general solvers that can be used to solve OT problems with a user-specified regularization.

Examples. The POT toolbox also provides several examples and notebooks which present various OT problems. Solvers are illustrated on one or two dimensional simulated data, as well as some real world problems such as color transfer.

4. Toolbox philosophy and underlying technology

In this section, we detail the vision of POT, and explain how we ensure the open source philosophy.

Code quality and library design. The main purpose of POT is to compute solutions to optimal transport problems such as the Wasserstein distance between probability distributions. To ensure code clarity, we respect the PEP8 guidance. Regarding code quality, we enforce a strict rule on providing several tests for every new contribution and detected bug. In some cases, OT problems have closed-form solutions which allow one to design simple tests and to ensure the correctness of our code. Through our test procedure, all functions and classes are tested, with a line coverage of 92% of the code. Optimal transport can be used to solve different families of problems and we have designed POT to provide a sub-module for each one of them. For instance, there are sub-modules for linear programming

solvers (`ot.lp`) or domain adaptation (`ot.da`). Some fundamental functions and solvers, such as `ot.emd`, are imported directly into the root `ot` module for easy access.

Open and collaborative. We have implemented POT with an open and collaborative spirit and placed it under MIT license. It is hosted on GitHub[1] and a public mailing list is available to the community to work on or follow the toolbox. POT is community-based, and has already received contributions from more than 20 collaborators. External contributions and requests are welcome and encouraged if they follow the code of conduct provided on the repository. For all contributions, continuous integration is used to ensure validation of all the tests and PEP8 compliance before any code merge (wheels compilation, tests, coverage, documentation build).

Documentation. POT provides detailed documentation[2] for each function and class. For each function, the documentation includes the mathematical problem that the function solves, parameter descriptions, returned elements, examples, and bibliographical references. All the papers related to the available solvers are listed in the documentation. Finally, we provide an easy to understand quick start guide and a gallery of examples in the documentation, both as Python scripts and Jupyter notebooks, thanks to the sphinx-gallery package. In total, more than 25 examples are available.

Dependencies. POT relies only on open source libraries such as `NumPy` (Harris et al., 2020) and `SciPy` (Virtanen et al., 2020) for linear algebra and optimization problems. Visualization, mostly used in the examples, requires the `Matplotlib` (Hunter, 2007) visualization library. The `Cython` (Behnel et al., 2011) framework was used to provide a simple Python wrapper around the C++ code. For some specific sub-modules such as dimensionality reduction with a Wasserstein loss, we used the manifold optimization library `Pymanopt` (Townsend et al., 2016) and the automatic differentiation library `Autograd` (Maclaurin et al., 2015). For solving the problem of non regularized OT, we have used a state-of-the-art solver provided in C++ by Bonneel et al. (2011). Finally, we allow for GPU acceleration with a `CuPy` implementation of an entropic OT; a `PyTorch` (Paszke et al., 2019) implementation for losses and solvers is currently underway.

5. Conclusion

We have presented POT, an optimal transport toolbox written in Python. Distributed under the MIT license, this open source toolbox is well-documented and community-driven. It offers reference implementations of state-of-the-art solvers for optimal transport related problems, and aims to foster the development and the fast dissemination of new ML algorithms. Code consistency makes it an easy tool for research purposes, teaching[3] and industrial applications. It relies solely on open source software. The toolbox is actively maintained and evolves quickly. Future directions for our toolbox include further integration possibilities with deep neural network pipelines, by providing native support for modern backends such as PyTorch (Paszke et al., 2019) or TensorFlow (Abadi et al., 2015), currently underway.

1. `https://github.com/PythonOT/POT`
2. `https://PythonOT.github.io`
3. POT was used as teaching tool for practical sessions at Statlearn 2018 and the DS3 Data Science Summer School 2019.

Acknowledgments

This work benefited from the support of the project OATMIL ANR-17-CE23-0012 and 3IA Cote d'Azur Investments ANR-19-P3IA-0002 of the French National Research Agency (ANR). This research was produced within the framework of Energy4Climate Interdisciplinary Center (E4C) of IP Paris and Ecole des Ponts ParisTech. This research was supported by 3rd Programme d'Investissements d'Avenir ANR-18-EUR-0006-02. Furthermore, we would like to thank Gabriel Peyré, Marco Cuturi and Nicolas Bonneel for providing code in various languages.

References

Martín Abadi, Ashish Agarwal, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Available from tensorflow.org.

Martial Agueh and Guillaume Carlier. Barycenters in the Wasserstein space. *SIAM Journal on Mathematical Analysis*, 43(2):904–924, 2011.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th ICML*, Proceedings of Machine Learning Research. PMLR, 2017.

Stefan Behnel, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn, and Kurt Smith. Cython: The best of both worlds. *Computing in Science & Engineering*, 13 (2):31–39, 2011.

Mathieu Blondel, Vivien Seguy, and Antoine Rolet. Smooth and sparse optimal transport. In *Proceedings of the Twenty-First AISTATS*, volume 84, pages 880–889. PMLR, 2018.

Nicolas Bonneel, Michiel van de Panne, Sylvain Paris, and Wolfgang Heidrich. Displacement interpolation using Lagrangian mass transport. *ACM Trans. Graph.*, 2011.

Nicolas Bonneel, Gabriel Peyré, and Marco Cuturi. Wasserstein barycentric coordinates: Histogram regression using optimal transport. *ACM Trans. Graph.*, 35(4):71:1–71:10, 2016.

Benjamin Charlier, Jean Feydy, Joan Alexis Glaunès, François-David Collin, and Ghislain Durif. Kernel operations on the gpu, with autodiff, without memory overflows. *arXiv preprint arXiv:2004.11127*, 2020.

Lenaïc Chizat. Numerical methods for (unbalanced) optimal transport, 2016. URL `https://github.com/lchizat/optimal-transport`.

Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2016.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, pages 2292–2300, 2013.

Marco Cuturi and David Avis. Ground metric learning. *J. Mach. Learn. Res.*, 15(1): 533–564, 2014.

Alban Farchi. Optimal transport problem solver, 2016. URL `http://cerea.enpc.fr/opttrans/`.

Jean Feydy, Pierre Roussillon, Alain Trouvé, and Pietro Gori. Fast and scalable optimal transport for brain tractograms. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 636–644. Springer, 2019.

Remi Flamary, Marco Cuturi, Nicolas Courty, and Alain Rakotomamonjy. Wasserstein discriminant analysis. *Machine Learning*, 2018.

Aude Genevay, Marco Cuturi, Gabriel Peyré, and Francis Bach. Stochastic optimization for large-scale optimal transport. In *Advances in Neural Information Processing Systems 29*, pages 3440–3448, 2016.

Alexandre Gramfort, Gabriel Peyré, and Marco Cuturi. Fast optimal transport averaging of neuroimaging data. In *IPMI*, pages 261–272, 2015.

Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95, 2007.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *ICML*, volume 37, pages 957–966, 2015.

Dougal Maclaurin, David Duvenaud, Matthew Johnson, and Ryan P. Adams. Autograd: Reverse-mode differentiation of native Python, 2015. URL `https://github.com/HIPS/autograd`.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, 2018.

Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vis.*, 40(2):99–121, 2000.

Bernard Schmitzer. Optimal transport toolbox 0.2, 2018. URL `https://github.com/bernhard-schmitzer/optimal-transport/`.

James Townsend, Niklas Koep, and Sebastian Weichwald. Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation. *Journal of Machine Learning Research*, 17(137):1–5, 2016.

Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: https://doi.org/10.1038/s41592-019-0686-2.

Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

Steven Zhang, David Widmann, and Davi Barreira. Optimal Transport in Julia, 2020. URL `http://zsteve.phatcode.net/OptimalTransportDocs/`.