

# Fast libraries for geometric data analysis

---

Jean Feydy  
HeKA team, Inria Paris  
Inserm, Université Paris-Cité

27th of February, 2023  
Machine Learning Coffee Seminar  
Finnish Center for Artificial Intelligence

# Who am I?

Background in **mathematics** and **data sciences**:

**2012–2016** ENS Paris, mathematics.

**2014–2015** M2 mathematics, vision, learning at ENS Cachan.

**2016–2019** PhD thesis in **medical imaging** with Alain Trouvé at ENS Cachan.

**2019–2021** **Geometric deep learning** with Michael Bronstein at Imperial College.

**2021+** **Medical data analysis** in the HeKA INRIA team (Paris).

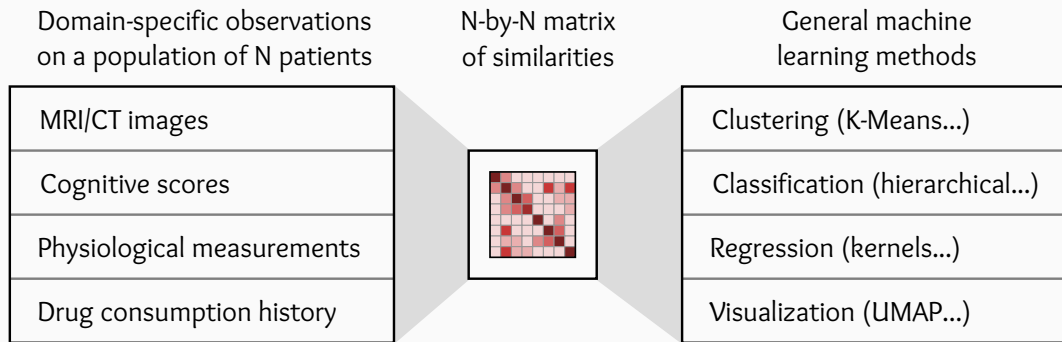
Close ties with **healthcare**:

**2015+** Medical imaging.

**2016+** Computational anatomy.

**2021+** Public health.

## A focus on the geometric side of data sciences



My research is about understanding **similarity structures**.

What are the implicit **priors** that they reflect?

How can we manipulate them **efficiently**?

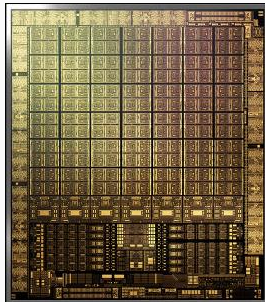
## A field that is moving fast

**Target.** Allow scientists to work with **tailor-made** models as **efficiently** as possible.

**Challenge.** The advent of **Graphics Processing Units (GPU)**:

- Incredible **value for money**:  
 $1\,000\text{€} \simeq 1\,000 \text{ cores} \simeq 10^{12} \text{ operations/s.}$
- **Bottleneck**: constraints on **register** usage.

“User-friendly” Python ecosystem, consolidated around  
a **small number of key operations**.



**7,000 cores**  
in a single GPU.

# My project: a long-term investment in the foundations of our field

**Solution.** Expand the **standard toolbox** in data sciences to deal with the challenges of the healthcare industry.

**Ease** the development of **advanced models**, without compromising on numerical performance.

Today's talk:

1. Efficient manipulation of “**symbolic**” **matrices** (distances, kernel, etc.).
2. **Optimal transport**: generalized sorting methods.
3. The long road to **standardization** and **clinical** impact.

# **1. Symbolic matrices**

---

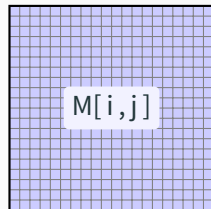
# Computing libraries represent most objects as tensors

**Context.** Constrained **memory accesses** on the GPU:

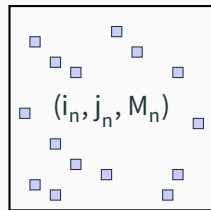
- **Long access times** to the registers penalize the use of large **dense** arrays.
- Hard-wired **contiguous** memory accesses penalize the use of **sparse** matrices.

**Challenge.** In order to reach optimal run times:

- **Restrict** ourselves to operations that are supported by the constructor: convolutions, FFT, etc.
- Develop new routines from scratch in C++/CUDA (FAISS, KPConv...): **several months of work.**



**Dense array**



**Sparse matrix**

# The KeOps library: efficient support for symbolic matrices

**Solution.** KeOps – [www.kernel-operations.io](http://www.kernel-operations.io):

- For PyTorch, NumPy, Matlab and R, on **CPU and GPU**.
- **Automatic differentiation**.
- Just-in-time **compilation** of **optimized** C++ schemes, triggered for every new **reduction**: sum, min, etc.

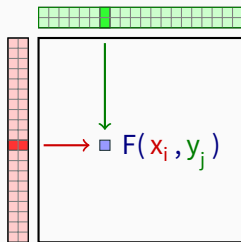
If the formula “F” is simple ( $\leq 100$  arithmetic operations):

“100k  $\times$  100k” computation  $\rightarrow$  10ms – 100ms,

“1M  $\times$  1M” computation  $\rightarrow$  1s – 10s.

Hardware ceiling of  $10^{12}$  operations/s.

$\times 10$  to  $\times 100$  **speed-up** vs standard GPU implementations  
for a wide range of problems.



## Symbolic matrix

Formula + data

- Distances  $d(x_i, y_j)$ .
- Kernel  $k(x_i, y_j)$ .
- Numerous transforms.



## A first example: efficient nearest neighbor search in dimension 50

Create large point clouds using **standard PyTorch syntax**:

```
import torch
N, M, D = 10**6, 10**6, 50
x = torch.rand(N, 1, D).cuda() # (1M, 1, 50) array
y = torch.rand(1, M, D).cuda() # (1, 1M, 50) array
```

Turn **dense** arrays into **symbolic** matrices:

```
from pykeops.torch import LazyTensor
x_i, y_j = LazyTensor(x), LazyTensor(y)
```

Create a large **symbolic matrix** of squared distances:

```
D_ij = ((x_i - y_j) ** 2).sum(dim=2) # (1M, 1M) symbolic
```

Use an `.argmin()` **reduction** to perform a nearest neighbor query:

```
indices_i = D_ij.argmin(dim=1) # -> standard torch tensor
```

## The KeOps library combines performance with flexibility

Script of the previous slide = efficient nearest neighbor query,  
**on par** with the bruteforce CUDA scheme of the **FAISS** library...

And can be used with **any metric**!

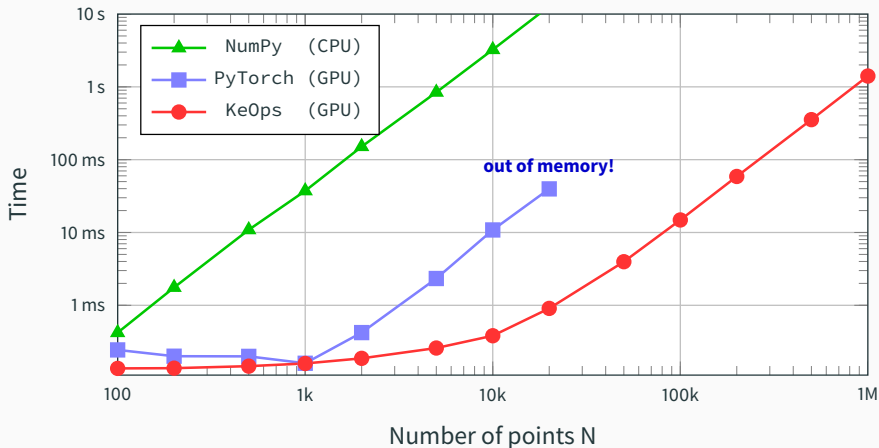
```
D_ij = ((x_i - x_j) ** 2).sum(dim=2)      # Euclidean  
M_ij = (x_i - x_j).abs().sum(dim=2)      # Manhattan  
C_ij = 1 - (x_i | x_j)                   # Cosine  
H_ij = D_ij / (x_i[...,0] * x_j[...,0])  # Hyperbolic
```

KeOps supports arbitrary **formulas** and **variables** with:

- **Reductions:** sum, log-sum-exp, K-min, matrix-vector product, etc.
- **Operations:** +, ×, sqrt, exp, neural networks, etc.
- **Advanced schemes:** batch processing, block sparsity, etc.
- **Automatic differentiation:** seamless integration with PyTorch.

# KeOps lets users work with millions of points at a time

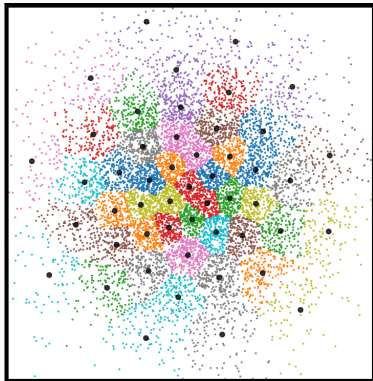
Benchmark of a Gaussian **convolution**  
between **clouds of N 3D points** on a RTX 2080 Ti GPU.



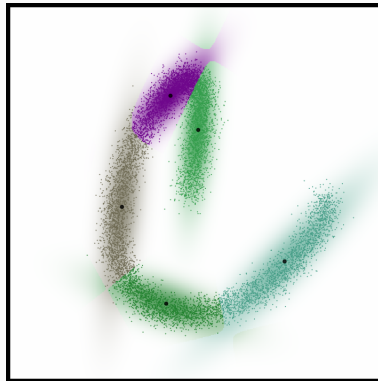
# Applications

---

## KeOps is a good fit for machine learning research



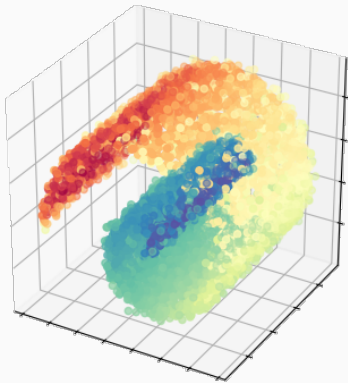
K-Means.



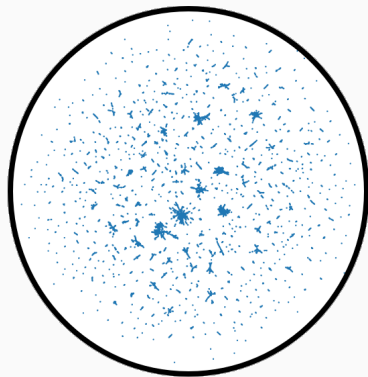
Gaussian Mixture Model.

Use **any** kernel, metric or formula **you** like!

## KeOps is a good fit for machine learning research



Spectral analysis.

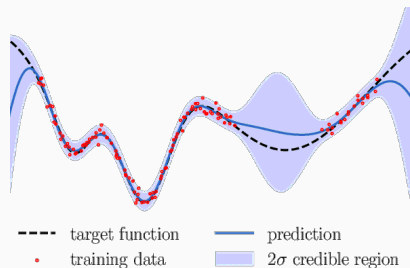


UMAP in hyperbolic space.

Use **any** kernel, metric or formula **you** like!

# Applications to Kriging, spline, Gaussian process, kernel regression

A standard tool for regression [Lec18]:



Under the hood, solve a **kernel linear system**:

$$(\lambda \text{Id} + K_{xx}) a = b \quad \text{i.e.} \quad a \leftarrow (\lambda \text{Id} + K_{xx})^{-1} b$$

where  $\lambda \geq 0$  et  $(K_{xx})_{i,j} = k(x_i, x_j)$  is a positive definite matrix.

**KeOps symbolic tensors**  $(K_{xx})_{i,j} = k(x_i, x_j)$  :

- Can be fed to **standard solvers**: SciPy, GPyTorch, etc.
- GPytorch on the 3DRoad dataset (N = 278k, D = 3):

**7h with 8 GPUs** → **15mn with 1 GPU.**

- Provide a **fast backend for research codes**:  
see e.g. *Kernel methods through the roof: handling **billions of points** efficiently*,  
by G. Meanti, L. Carratino, L. Rosasco, A. Rudi (2020).



# Geometric deep learning

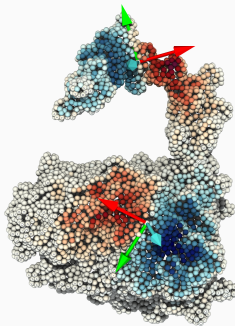
**Context.** Trainable models on **non-Euclidean domains** (point clouds, surfaces, graphs, etc.), beyond 2D/3D images.

**Challenge.** In spite of growing interest in the industry, these models still **lack support** on the numerical side. C++/CUDA is (often) required to reach top performance.

**Solution.** Using KeOps, with a few lines of Python:

- **Local** interactions: K-nearest neighbors.
- **Global** interactions: generalized convolutions.

Modelling **freedom**  
⇒ **Domain-specific** priors.



Quasi-geodesic  
convolution on a  
protein surface.

## **2. Fast optimal transport solvers**

---

# Optimal transport (OT) generalizes sorting to spaces of dimension $D > 1$

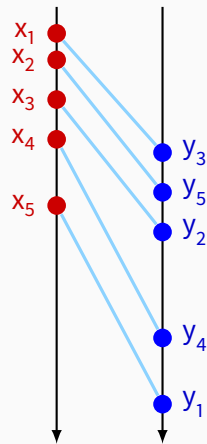
**Context.** If  $A = (x_1, \dots, x_N)$  and  $B = (y_1, \dots, y_N)$  are two clouds of  $N$  points in  $\mathbb{R}^D$ , we define:

$$\text{OT}(A, B) = \min_{\sigma \in \mathcal{S}_N} \frac{1}{2N} \sum_{i=1}^N \|x_i - y_{\sigma(i)}\|^2$$

Generalizes **sorting** to metric spaces.

We turn a **distance matrix** into a **permutation**.

We extend this definition to **weighted** samples, **continuous** distributions with **outliers**, etc.



# Optimal transport has two main uses in data sciences

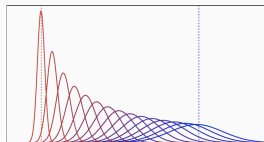
The **optimal matching**  $x_i \mapsto y_{\sigma(i)}$  is:

- A **nearest neighbor** projection subject to a **bijectivity** constraint.
- A fundamental operation in 3D shape analysis.
- A staple of operations research.

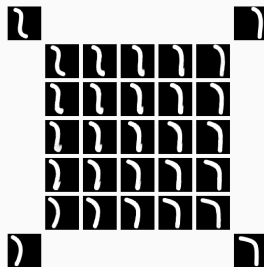
The **total cost**  $\text{OT}(\mathbf{A}, \mathbf{B})$  induces:

- A useful **distance** between probability distributions.
- Particle-based **interpolation** with

$$\arg \min_{\mathbf{A}} \lambda_1 \text{OT}(\mathbf{A}, \mathbf{B}_1) + \dots + \lambda_K \text{OT}(\mathbf{A}, \mathbf{B}_K).$$



OT geodesic



OT barycenters

## But how should we solve the OT problem?

Key dates for discrete optimal transport with  $N$  points:

- [Kan42]: **Dual** problem of Kantorovitch.
- [Kuh55]: **Hungarian** methods in  $O(N^3)$ .
- [Ber79]: **Auction** algorithm in  $O(N^2)$ .
- [KY94]: **SoftAssign** = Sinkhorn + simulated annealing, in  $O(N^2)$ .
- [GRL<sup>+</sup>98, CR00]: **Robust Point Matching** = Sinkhorn as a loss.
- [Cut13]: Start of the **GPU era**.
- [Mér11, Lév15, Sch19]: **multi-scale** solvers in  $O(N \log N)$ .
- **Solution**, today: **Multiscale Sinkhorn algorithm, on the GPU**.  
 $\implies$  Generalized **QuickSort** algorithm.

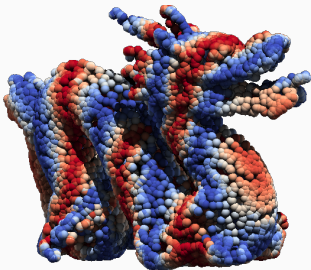
# Scaling up optimal transport to anatomical data

Progresses of the last decade add up to a  $\times 100$  -  $\times 1000$  acceleration:

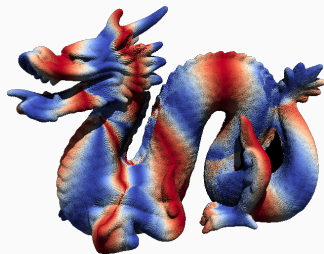
Sinkhorn GPU  $\xrightarrow{\times 10}$  + KeOps  $\xrightarrow{\times 10}$  + Annealing  $\xrightarrow{\times 10}$  + Multi-scale

With a precision of 1%, on a modern gaming GPU:

`pip install  
geomloss`  
+  
modern GPU  
(1 000 €)



10k points in 30-50ms



**100k points in 100-200ms**

## Conclusion

---

## Key points

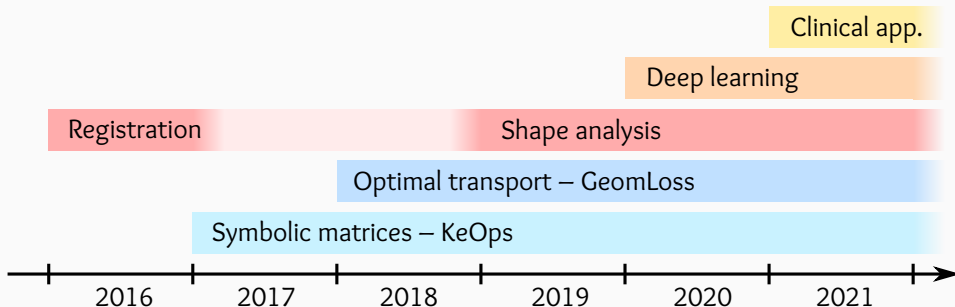
- **Symbolic** matrices are to **geometric** ML what **sparse** matrices are to **graph** processing:
  - KeOps: **x30 speed-up** vs. PyTorch, TF et JAX.
  - Useful in a wide range of settings.
- Optimal Transport = **generalized sorting** :
  - Simple registration for shapes that are close to each other.
  - Super-fast  $O(N \log N)$  solvers.
- These tools open **new paths** for geometers and statisticians:
  - GPUs are more **versatile** than you think.
  - Ongoing work to provide **fast GPU backends** to researchers, going beyond what Google and Facebook are ready to pay for.



## Summary: a long-term investment that is starting to bear fruits

Two major evolutions:

- “Big” geometric problem:  $N > 10k \rightarrow N > 1M$ .
- Optimal transport: linear **problem** + generalized **quicksort**.



## Genuine team work



Alain Trouvé



Thibault Séjourné



F.-X. Vialard



Gabriel Peyré



Benjamin Charlier



Joan Glaunès



Freyr Sverrisson



Shen Zhengyang

+ Marc Niethammer, Bruno Correia, Michael Bronstein...

## Going forward: the long road to genuine clinical impact

These tools are diffusing well in our research communities (400k+ downloads).

The target is now to **go beyond “expert users”**.

First step in March 2022: removed all problematic **dependencies** from KeOps 2.0.

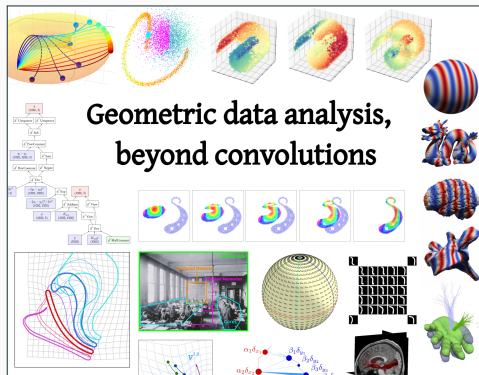
We are now working on:

- High performance on **CPU**.
- A 100% transparent and NumPy-compatible **API** for KeOps+GeomLoss.
- Standard **benchmarks** for kernel methods and optimal transport.
- Applications to **drug consumption** data from the SNDS  
with Anne-Sophie Jannot, Alexis Van Straaten and Pierre Sabatier.

I hope that we'll have nice results to show you soon :-)

**Documentation and tutorials are available online**

[www.kernel-operations.io](http://www.kernel-operations.io)



[www.jeanfeydy.com/geometric\\_data\\_analysis.pdf](http://www.jeanfeydy.com/geometric_data_analysis.pdf)

[www.jeanfeydy.com/Teaching](http://www.jeanfeydy.com/Teaching)

## References

---



Dimitri P Bertsekas.

**A distributed algorithm for the assignment problem.**

*Lab. for Information and Decision Systems Working Paper, M.I.T., Cambridge, MA, 1979.*



Grégoire Clarté, Antoine Diez, and Jean Feydy.

**Collective proposal distributions for nonlinear MCMC samplers: Mean-field theory and fast implementation.**

*arXiv preprint arXiv:1909.08988, 2019.*



Christophe Chnafa, Simon Mendez, and Franck Nicoud.

**Image-based large-eddy simulation in a realistic left heart.**



*Computers & Fluids*, 94:173–187, 2014.



Haili Chui and Anand Rangarajan.

**A new algorithm for non-rigid point matching.**

*In Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 44–51. IEEE, 2000.

-  Adam Conner-Simons and Rachel Gordon.  
**Using ai to predict breast cancer and personalize care.**  
<http://news.mit.edu/2019/using-ai-predict-breast-cancer-and-personalize-care-0507>, 2019.  
MIT CSAIL.
-  Marco Cuturi.  
**Sinkhorn distances: Lightspeed computation of optimal transport.**  
*In Advances in Neural Information Processing Systems*, pages 2292–2300, 2013.



 Pierre Degond, Amic Frouvelle, Sara Merino-Aceituno, and Ariane Trescases.

**Alignment of self-propelled rigid bodies: from particle systems to macroscopic equations.**

In *International workshop on Stochastic Dynamics out of Equilibrium*, pages 28–66. Springer, 2017.

 Pierre Degond and Sébastien Motsch.

**Continuum limit of self-driven particles with orientation interaction.**

*Mathematical Models and Methods in Applied Sciences*, 18(supp01):1193–1215, 2008.

 Steven Gold, Anand Rangarajan, Chien-Ping Lu, Suguna Pappu, and Eric Mjolsness.

**New algorithms for 2d and 3d point matching: Pose estimation and correspondence.**

*Pattern recognition*, 31(8):1019–1031, 1998.

 Leonid V Kantorovich.

**On the translocation of masses.**

In *Dokl. Akad. Nauk. USSR (NS)*, volume 37, pages 199–201, 1942.



Harold W Kuhn.

**The Hungarian method for the assignment problem.**

*Naval research logistics quarterly*, 2(1-2):83–97, 1955.



Jeffrey J Kosowsky and Alan L Yuille.

**The invisible hand algorithm: Solving the assignment problem with statistical physics.**

*Neural networks*, 7(3):477–490, 1994.



Florent Leclercq.

**Bayesian optimization for likelihood-free cosmological inference.**


*Physical Review D*, 98(6):063511, 2018.



Bruno Lévy.

**A numerical algorithm for l2 semi-discrete optimal transport in 3d.**

*ESAIM: Mathematical Modelling and Numerical Analysis*, 49(6):1693–1715, 2015.

 Christian Ledig, Andreas Schuh, Ricardo Guerrero, Rolf A Heckemann, and Daniel Rueckert.

**Structural brain imaging in Alzheimer's disease and mild cognitive impairment: biomarker analysis and shared morphometry database.**

*Scientific reports*, 8(1):11258, 2018.

 Quentin Mérigot.

**A multiscale approach to optimal transport.**

In *Computer Graphics Forum*, volume 30, pages 1583–1592. Wiley Online Library, 2011.



Bernhard Schmitzer.

**Stabilized sparse scaling algorithms for entropy regularized transport problems.**

*SIAM Journal on Scientific Computing*, 41(3):A1443–A1481, 2019.



Freyr Sverrisson, Jean Feydy, Bruno E. Correia, and Michael M. Bronstein.

**Fast end-to-end learning on protein surfaces.**

*bioRxiv*, 2020.