Fast libraries for geometric data analysis

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

22nd of May, 2023
Healthcare AI Grand Round
Nvidia, online
Who am I?

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

- **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

Domain-specific observations on a population of N patients

MRI/CT images
Cognitive scores
Physiological measurements
Drug consumption history

N-by-N matrix of similarities

General machine learning methods

Clustering (K-Means...)
Classification (hierarchical...)
Regression (kernels...)
Visualization (UMAP...)

My research is about understanding **similarity structures**. What are the implicit **priors** that they reflect? How can we manipulate them **efficiently**?
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:
  \[1000\text{€} \approx 1000 \text{ cores} \approx 10^{12} \text{ operations/s}.
- Bottleneck: low-level memory usage.

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

7,000 cores in a single GPU.
**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**.
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 (≤ 100 arithmetic operations):

- “100k × 100k” computation → 10ms – 100ms,
- “1M × 1M” computation → 1s – 10s.

Hardware ceiling of 10^{12} operations/s.
×10 to ×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**:

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

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

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

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

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

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

```python
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 brute force CUDA scheme of the **FAISS** library…

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

\[
D_{ij} = ((x_i - x_j)^2).sum(dim=2) \quad \# \text{ Euclidean}
\]

\[
M_{ij} = (x_i - x_j).abs().sum(dim=2) \quad \# \text{ Manhattan}
\]

\[
C_{ij} = 1 - (x_i \mid x_j) \quad \# \text{ Cosine}
\]

\[
H_{ij} = D_{ij} / (x_i[...,
0] * x_j[...,
0]) \quad \# \text{ Hyperbolic}
\]

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  

\[ a_i \leftarrow \sum_{j=1}^{N} \exp(-\|x_i - y_j\|_2^2) b_j \]

between clouds of N 3D points on a A100 GPU.
Yet another ML compiler?

Many impressive tools out there (Numba, Triton, Halide…):

- Focus on **generality** (software + hardware).
- Increasingly easy to use via e.g. PyTorch 2.0.

KeOps fills a different niche (a bit like cuFFT, FFTW…):

- Focus on a **single major bottleneck**: geometric interactions.
- **Agnostic** with respect to Euclidean / non-Euclidean formulas.
- Fully compatible with PyTorch, NumPy, R.
- Can actually be **used by mathematicians**.

KeOps is a **bridge** between geometers (with a maths background) and compiler experts (with a CS background).
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**:

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

where \(\lambda \geq 0\) et \((K_{xx})_{i,j} = k(x_i, x_j)\) is a positive definite matrix.
Applications to Kriging, spline, Gaussian process, kernel regression

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 \( \rightarrow \) 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.
Applications to protein sciences [SFCB20]

(a) Raw protein data.

(b) Interface.

(c) Prediction.
Fast end-to-end learning on protein surfaces

1. MaSIF
   a. surface mesh
   b. features
   c. patches
   d. output
   6 s → 20 s → 50 s → 165 ms

2. Ours
   a. points, normals
   b. features
   c. coordinates
   d. output
   70 ms → 50 ms → 6 ms → 40 ms

Pre-processing on the fly → ×100-×1,000 faster, lighter and fully differentiable.
Idea 1: on-the-fly sampling of protein surfaces

Fast, fully differentiable, heterogeneous batches (without padding).
Idea 2: quasi-geodesic convolutions

Fast, fully differentiable, heterogeneous batches (without padding).
KeOps lets us implement:

- **Custom** operations that best reflect a biological prior.
- Zero need to talk about CUDA blocks, threads, etc.
- Great tool for prototyping with geometric ideas.

**Main limitation:** beyond 16-32 channels per convolution, register spilling.

This is just **one example** of architecture that is equivariant to isometries.

(Some?) general E3NN layers could also be accelerated: we can talk about it.
2. Fast optimal transport solvers
Optimal transport (OT) generalizes sorting to spaces of dimension $D > 1$

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

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

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.

assignment
$\sigma : [1, 5] \rightarrow [1, 5]$
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 $\text{OT}(A, B)$ induces:
- A useful distance between probability distributions.
- Particle-based interpolation with
  $$\arg\min_A \lambda_1 \text{OT}(A, B_1) + \cdots + \lambda_k \text{OT}(A, B_k).$$
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+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
```

moderate GPU (1 000 €)

10k points in 30-50ms

100k points in 100-200ms
Lung registration “Exhale – Inhale”

Complex deformations, high resolution (50k–300k points), high accuracy (< 1mm).
State-of-the-art networks – and their limitations

Point neural nets, **in practice:**
- Compute **descriptors** at all scales.
- **Match** them using geometric layers.
- Train on **synthetic** deformations.

**Strengths and weaknesses:**
- Good at **pairing** branches.
- Hard to train to high **accuracy**.

⇒ **Complementary** to OT.

---

Multi-scale convolutional point neural network.
Three-steps registration

1. Affine-RobOT pre-alignment.
2.a. Deep prediction network.
2.b. Smooth deformation model.

This **pragmatic** method:

- Is **easy to train** on synthetic data.
- Scales up to high-resolution: 100k points in 1s.
- Excellent results: **KITTl** (outdoors scans) and **DirLab** (lungs).

Three-steps registration

0. Input data
1. Pre-alignment

Zoom!
2. Deep registration
3. Fine-tuning
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 people 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 (500k+ 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 70M French people 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

www.jeanfeydy.com/geometric_data_analysis.pdf
www.jeanfeydy.com/Teaching
References
Dimitri P Bertsekas.

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


Grégoire Clarté, Antoine Diez, and Jean Feydy.

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

Christophe Chnafa, Simon Mendez, and Franck Nicoud.

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


Haili Chui and Anand Rangarajan.

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

Adam Conner-Simons and Rachel Gordon.

Using ai to predict breast cancer and personalize care.


MIT CSAIL.

Marco Cuturi.

Sinkhorn distances: Lightspeed computation of optimal transport.

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

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


Pierre Degond and Sébastien Motsch.

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

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

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


Leonid V Kantorovich.

On the translocation of masses.

Harold W Kuhn.

The Hungarian method for the assignment problem.


Jeffrey J Kosowsky and Alan L Yuille.

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

Florent Leclercq.

Bayesian optimization for likelihood-free cosmological inference.


Bruno Lévy.

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

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.**


Quentin Mérigot.

**A multiscale approach to optimal transport.**

Bernhard Schmitzer.

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


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

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