Robust shape matching with Optimal Transport

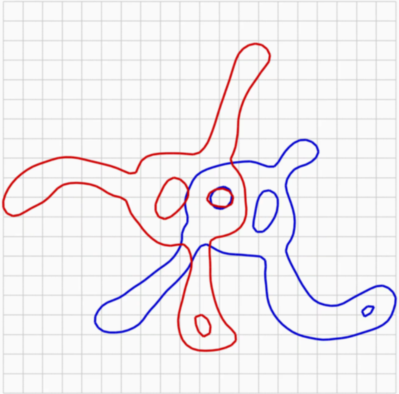
Jean Feydy

Télécom Paristech – 15th November, 2018

Écoles Normales Supérieures de Paris et Paris-Saclay
Collaboration with B. Charlier, J. Glaunès (KeOps library);
S.-i. Amari, G. Peyré, T. Séjourné, A. Trouvé, F.-X. Vialard (OT theory)

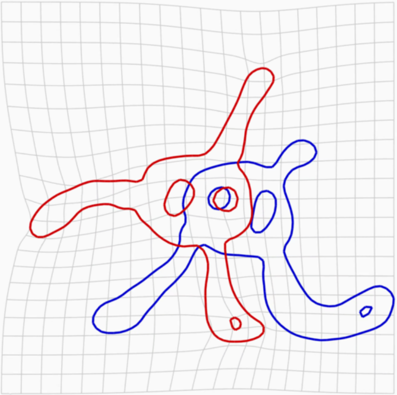
What is shape matching?

Source A, target B,



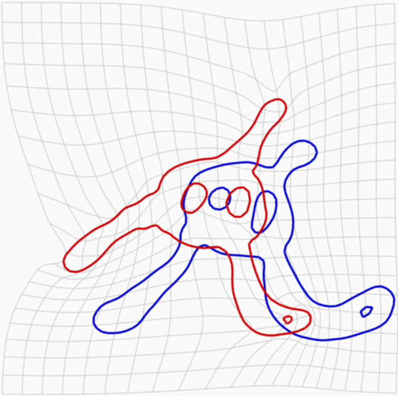
What is shape matching?

Source A, target B, mapping φ



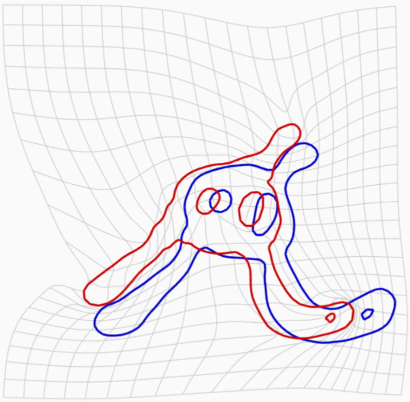
What is shape matching?

Source A, target B, mapping φ



What is shape matching?

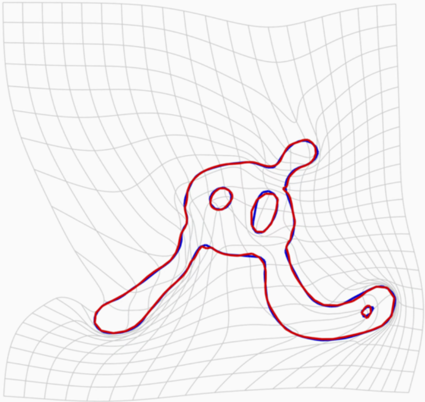
Source A, target B, mapping φ



What is shape matching?

Source A, target B, mapping φ

$$A \xrightarrow[\text{Model}]{\varphi} \varphi(A) = A' \underset{\textbf{Loss}}{\rightleftarrows} B$$



A good Loss function is a guarantee of robustness

Iterative Matching Algorithm

$A' \leftarrow A$
repeat
  $L\,,\ v \ \leftarrow\ \text{Loss}(A', B)\,,\ \ -\partial_{A'}\text{Loss}(A', B)$
  $A' \ \leftarrow\ A' \ +\ \text{Model}(\,v\,)$
until $\ L \ <\ \text{tol}$
Output : deformed shape $\ A' \ =\ \varphi(\,A\,)$.

A good Loss function is a guarantee of robustness

Iterative Matching Algorithm

1: $A' \leftarrow A$
2: **repeat**
3: $\quad L\,,\ v \leftarrow \text{Loss}(A', B)\,,\ -\partial_{A'}\text{Loss}(A', B)$
4: $\quad A' \leftarrow A' + \text{Model}(\,v\,)$
5: **until** $L < \text{tol}$
Output : deformed shape $A' = \varphi(\,A\,)$.

"Model" encodes the **prior knowledge** on admissible deformations:

- *smoothing* convolution
- LDDMM/SVF *backprop* + regularization + *shooting*
- *trained* neural network

A good Loss function is a guarantee of robustness

Iterative Matching Algorithm

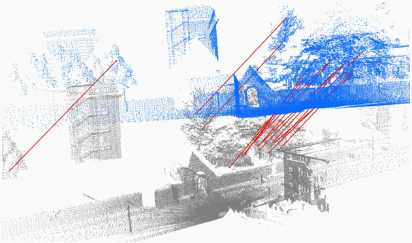
1: $A' \leftarrow A$
2: **repeat**
3: $\quad L\,,\, v \;\leftarrow\; \text{Loss}(A', B)\,,\;\; -\partial_{A'}\text{Loss}(A', B)$
4: $\quad A' \;\leftarrow\; A' \,+\, \text{Model}(\,v\,)$
5: **until** $\;L \,<\, \text{tol}$
Output : deformed shape $\;A' \;=\; \varphi(\,A\,)$.

"Model" encodes the **prior knowledge** on admissible deformations:

- *smoothing* convolution
- LDDMM/SVF *backprop* + regularization + *shooting*
- *trained* neural network

$\Rightarrow$ The *raw* Loss gradient v is what **drives** the registration

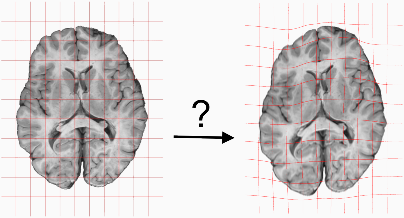
First setting: processing of point clouds



From the documentation of the Point Cloud Library.

- φ is **rigid** or affine
- Occlusions
- Outliers

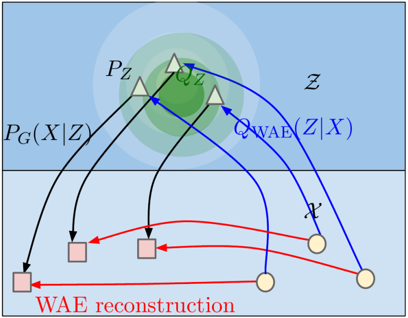
Second setting: medical imaging



From Marc Niethammer's Quicksilver slides.

- φ is a spline or a **diffeomorphism**
- Ill-posed problem
- Some occlusions

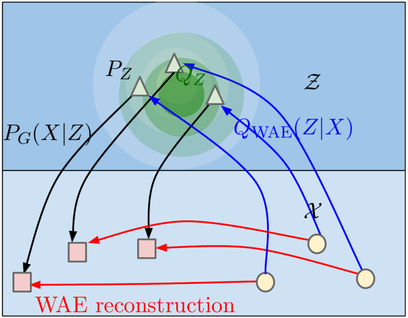
Third setting: training a generative model



Wasserstein Auto-Encoders,
Tolstikhin et al., 2018.

- φ is a **neural network**
- Very weak regularization
- High-dimensional space

Third setting: training a generative model

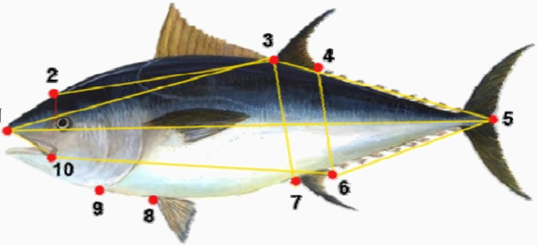


Wasserstein Auto-Encoders,
Tolstikhin et al., 2018.

- φ is a **neural network**
- Very weak regularization
- High-dimensional space

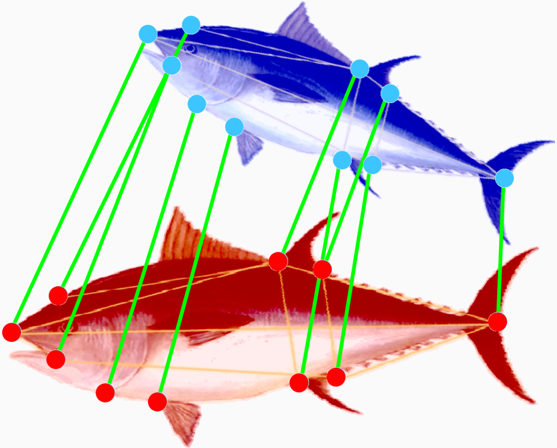
Which **Loss** function should we use?

On labeled shapes, use a spring energy



Anatomical landmarks from *A morphometric approach for the analysis of body shape in bluefin tuna*, Addis et al., 2009.

On labeled shapes, use a spring energy



Anatomical landmarks from *A morphometric approach for the analysis of body shape in bluefin tuna*, Addis et al., 2009.

Encoding unlabeled shapes as measures

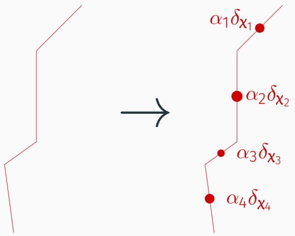
Let's enforce sampling invariance:

$$A \longrightarrow \alpha = \sum_{i=1}^{N} \alpha_i \delta_{x_i}, \qquad B \longrightarrow \beta = \sum_{j=1}^{M} \beta_j \delta_{y_j}.$$

Encoding unlabeled shapes as measures

Let's enforce sampling invariance:

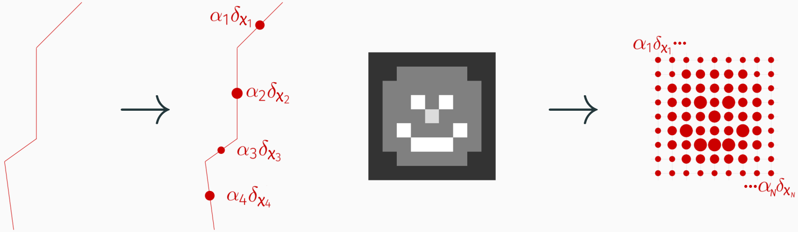
$$A \longrightarrow \alpha = \sum_{i=1}^{N} \alpha_i \delta_{x_i}, \qquad B \longrightarrow \beta = \sum_{j=1}^{M} \beta_j \delta_{y_j}.$$



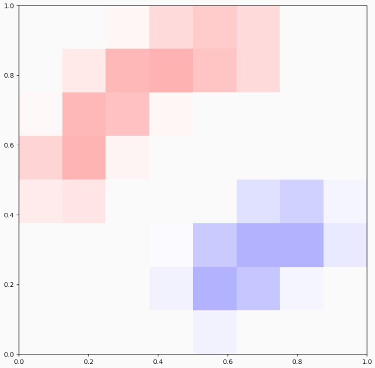
Encoding unlabeled shapes as measures

Let's enforce sampling invariance:

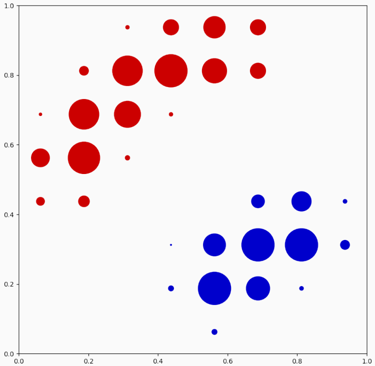
$$A \longrightarrow \alpha = \sum_{i=1}^{N} \alpha_i \delta_{x_i}, \qquad B \longrightarrow \beta = \sum_{j=1}^{M} \beta_j \delta_{y_j}.$$



A baseline setting: density registration

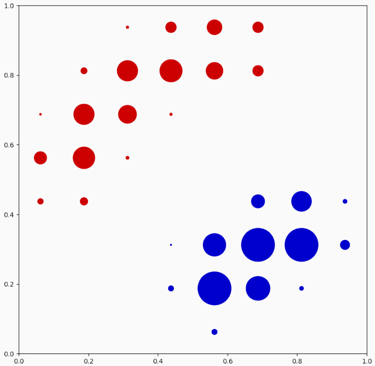


A baseline setting: density registration



$$\alpha = \sum_{i=1}^{N} \alpha_i \delta_{x_i}\,, \quad \beta = \sum_{j=1}^{M} \beta_j \delta_{y_j}\,.$$

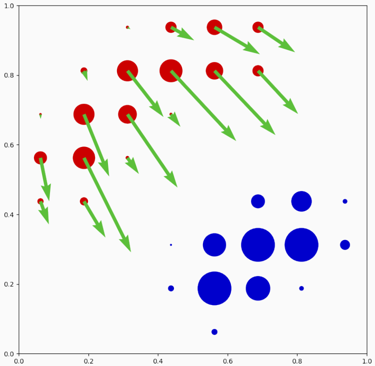
A baseline setting: density registration



$$\alpha = \sum_{i=1}^{N} \alpha_i \delta_{x_i}\,, \quad \beta = \sum_{j=1}^{M} \beta_j \delta_{y_j}\,.$$

$$\sum_{i=1}^{N} \alpha_i \;=\; 1 \;=\; \sum_{j=1}^{M} \beta_j$$

A baseline setting: density registration

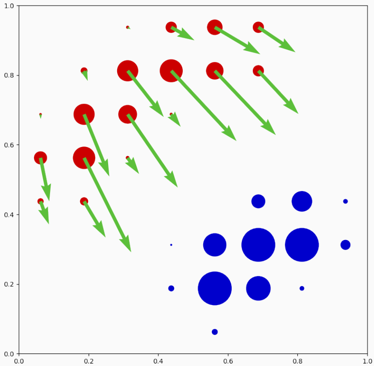


$$\alpha = \sum_{i=1}^{N} \alpha_i \delta_{x_i}\,, \quad \beta = \sum_{j=1}^{M} \beta_j \delta_{y_j}\,.$$

$$\sum_{i=1}^{N} \alpha_i \;=\; 1 \;=\; \sum_{j=1}^{M} \beta_j$$

Display $v \;=\; -\nabla_{x_i} \mathrm{d}(\alpha, \beta)$.

A baseline setting: density registration



$$\alpha = \sum_{i=1}^{N} \alpha_i \delta_{x_i}, \quad \beta = \sum_{j=1}^{M} \beta_j \delta_{y_j}.$$

$$\sum_{i=1}^{N} \alpha_i = 1 = \sum_{j=1}^{M} \beta_j$$

Display $\quad v = -\nabla_{x_i} \mathrm{d}(\alpha, \beta)$.

Seamless extensions to:

- $\sum_i \alpha_i \neq \sum_j \beta_j$, outliers [Chizat et al., 2018],
- curves and surfaces [Kaltenmark et al., 2017],
- variable weights α_i.

Overview

Computing fidelities between **measures**:

1. **Computer graphics**: weighted Hausdorff distance
2. **Statistics**: kernel distances
3. **Optimal Transport**: Wasserstein distance

   $\simeq$ Robust Point Matching

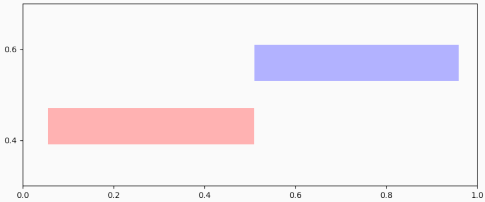
Overview

Computing fidelities between **measures**:

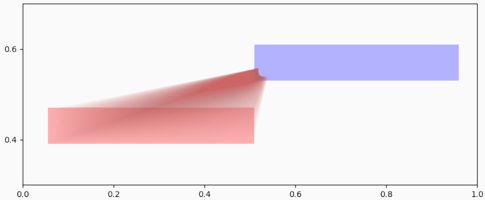
1. **Computer graphics**: weighted Hausdorff distance
2. **Statistics**: kernel distances
3. **Optimal Transport**: Wasserstein distance
   $\simeq$ Robust Point Matching
4. What's **new**, in 2018?
5. Efficient GPU routines: **KeOps**

The weighted Hausdorff distance: Iterative Closest Point algorithm

The weighted Hausdorff distance



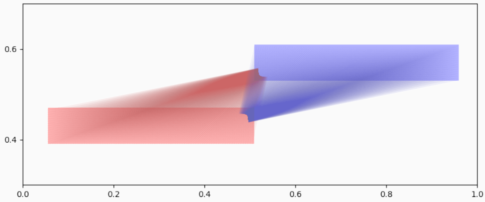
The weighted Hausdorff distance



p-Hausdorff distance:

$$\text{Loss}(\alpha, \beta) \quad = \quad \tfrac{1}{2} \sum_i \alpha_i \cdot \min_j \|x_i - y_j\|^p$$

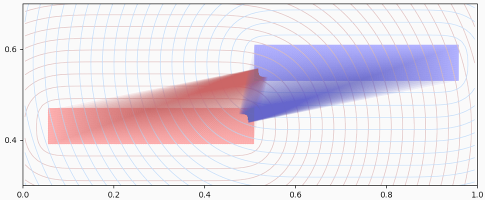
The weighted Hausdorff distance



p-Hausdorff distance:

$$\text{Loss}(\alpha, \beta) \;=\; \tfrac{1}{2}\sum_i \alpha_i \cdot \min_j \|x_i - y_j\|^p \;+\; \tfrac{1}{2}\sum_j \beta_j \cdot \min_i \|x_i - y_j\|^p$$

The weighted Hausdorff distance

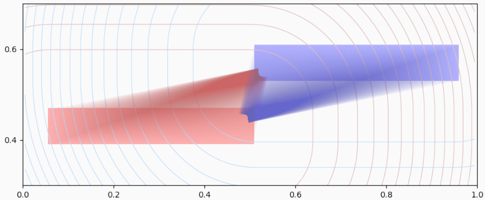


p-Hausdorff distance:

$$\begin{aligned} \text{Loss}(\alpha, \beta) &= \tfrac{1}{2} \textstyle\sum_i \alpha_i \cdot \min_j \|x_i - y_j\|^p &+& \tfrac{1}{2} \textstyle\sum_j \beta_j \cdot \min_i \|x_i - y_j\|^p \\ &= \tfrac{1}{2} \langle\, \alpha \,,\, b \,\rangle &+& \tfrac{1}{2} \langle\, \beta \,,\, a \,\rangle \end{aligned}$$

with
$$\begin{aligned} a(x) &= \mathrm{d}(\, x \,,\, \mathrm{supp}(\alpha))^p \\ b(x) &= \mathrm{d}(\, x \,,\, \mathrm{supp}(\beta))^p \end{aligned}$$

The weighted Hausdorff distance

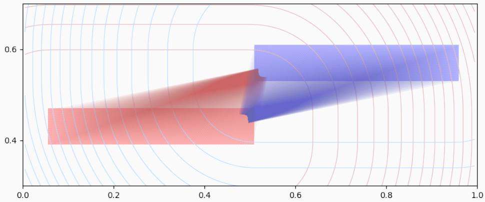


p-Hausdorff distance:

$$\begin{aligned}\text{Loss}(\alpha,\beta) &= \tfrac{1}{2}\sum_i \alpha_i \cdot \min_j \|x_i - y_j\|^p &+& \tfrac{1}{2}\sum_j \beta_j \cdot \min_i \|x_i - y_j\|^p \\ &= \tfrac{1}{2}\langle\, \alpha \,,\, b \,\rangle &+& \tfrac{1}{2}\langle\, \beta \,,\, a \,\rangle\end{aligned}$$

$$\text{with} \qquad \begin{aligned} a(x) &= \mathrm{d}(\,x\,,\,\mathrm{supp}(\alpha))^p \\ b(x) &= \mathrm{d}(\,x\,,\,\mathrm{supp}(\beta))^p \end{aligned}$$

The weighted Hausdorff distance

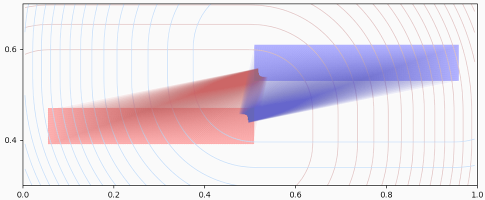


p-Hausdorff distance:

$$\begin{aligned}\text{Loss}(\alpha,\beta) \;&=\; \tfrac{1}{2}\textstyle\sum_i \alpha_i \cdot \min_j \|x_i - y_j\|^p \;&+\;& \tfrac{1}{2}\textstyle\sum_j \beta_j \cdot \min_i \|x_i - y_j\|^p \\ &=\; \tfrac{1}{2}\langle\, \alpha \,,\, b \,\rangle &+\;& \tfrac{1}{2}\langle\, \beta \,,\, a \,\rangle \\ &=\; \tfrac{1}{2}\langle\, \alpha \,,\, b - a \,\rangle &+\;& \tfrac{1}{2}\langle\, \beta \,,\, a - b \,\rangle\end{aligned}$$

$$\text{with} \qquad \begin{aligned} a(x) &= \mathrm{d}(\,x\,,\, \mathrm{supp}(\alpha))^p \\ b(x) &= \mathrm{d}(\,x\,,\, \mathrm{supp}(\beta))^p \end{aligned}$$

The weighted Hausdorff distance



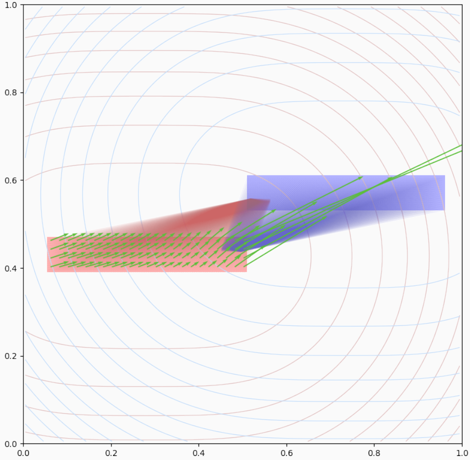
p-Hausdorff distance:

$$\begin{aligned}
\mathrm{Loss}(\alpha,\beta) &= \tfrac{1}{2}\textstyle\sum_i \alpha_i \cdot \min_j \|x_i - y_j\|^p &&+ \tfrac{1}{2}\textstyle\sum_j \beta_j \cdot \min_i \|x_i - y_j\|^p \\
&= \tfrac{1}{2}\langle\, \alpha \,,\, b \,\rangle &&+ \tfrac{1}{2}\langle\, \beta \,,\, a \,\rangle \\
&= \tfrac{1}{2}\langle\, \alpha \,,\, b - a \,\rangle &&+ \tfrac{1}{2}\langle\, \beta \,,\, a - b \,\rangle \\
&= \tfrac{1}{2}\langle\, \alpha - \beta \,,\, b - a \,\rangle
\end{aligned}$$

$$\text{with} \qquad \begin{aligned} a(x) &= \mathrm{d}(\,x\,,\,\mathrm{supp}(\alpha))^p \\ b(x) &= \mathrm{d}(\,x\,,\,\mathrm{supp}(\beta))^p \end{aligned}$$

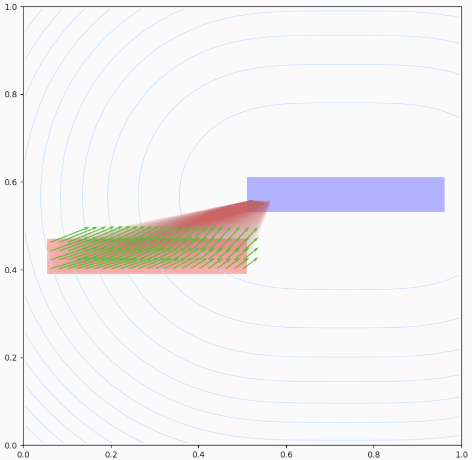
Naive projections in Hausdorff cause imbalance

$$\text{Loss}(\alpha, \beta) \quad = \quad \tfrac{1}{2}\langle\, \alpha,\ b - a \,\rangle \;+\; \tfrac{1}{2}\langle\, \beta,\ a - b \,\rangle$$



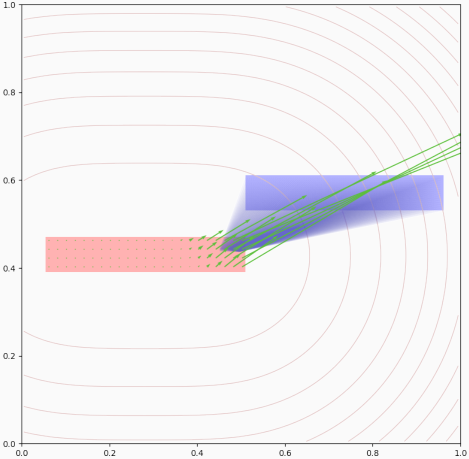
Naive projections in Hausdorff cause imbalance

$$\text{Loss}(\alpha, \beta) \quad = \quad \tfrac{1}{2}\langle\, \alpha,\ b - a\,\rangle \ + \ \tfrac{1}{2}\langle\, \beta,\ a - b\,\rangle$$



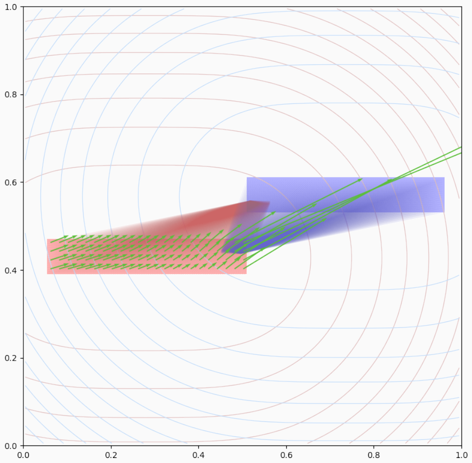
Naive projections in Hausdorff cause imbalance

$$\text{Loss}(\alpha, \beta) \quad = \quad \tfrac{1}{2}\langle\, \alpha,\ b - a \,\rangle \ + \ \tfrac{1}{2}\langle\, \beta,\ a - b \,\rangle$$



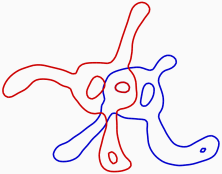
Naive projections in Hausdorff cause imbalance

$$\text{Loss}(\alpha, \beta) \quad = \quad \tfrac{1}{2}\langle\, \alpha,\ b - a\,\rangle \ + \ \tfrac{1}{2}\langle\, \beta,\ a - b\,\rangle$$



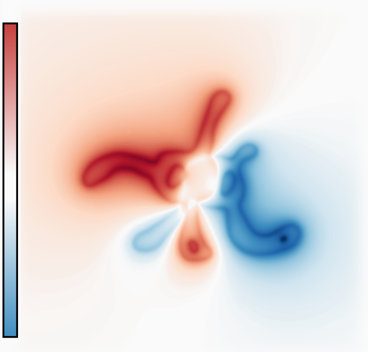
An idea from statistics: Kernel distances

Kernel fidelities: the simplest formula for $d(\alpha, \beta)$



Raw signal $(\alpha - \beta)$.

Kernel fidelities: the simplest formula for $d(\alpha, \beta)$

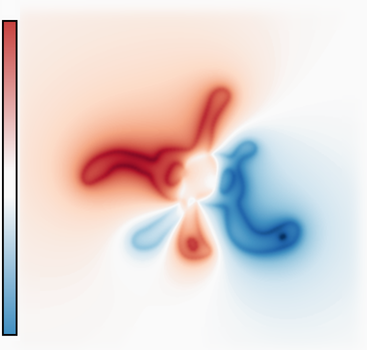


Blurred signal $g \star (\alpha - \beta)$.

Choose a symmetric blurring function g, a **kernel** $k = g \star g$:

$$\mathrm{d}_k(\alpha, \beta) \;=\; \| \, g \star \alpha - g \star \beta \, \|_{L^2}^2$$

Kernel fidelities: the simplest formula for $d(\alpha, \beta)$

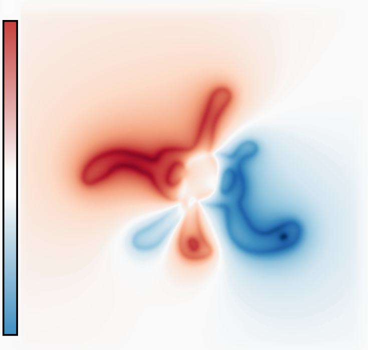


Blurred signal $g \star (\alpha - \beta)$.

Choose a symmetric blurring function g, a **kernel** $k = g \star g$:

$$\begin{aligned} \mathrm{d}_k(\alpha, \beta) &= \| g \star \alpha - g \star \beta \|_{L^2}^2 \\ &= \langle \alpha - \beta \,|\, k \star (\alpha - \beta) \rangle \end{aligned}$$

Kernel fidelities: the simplest formula for $d(\alpha, \beta)$

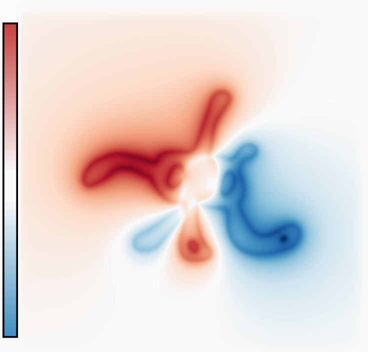


Blurred signal $g \star (\alpha - \beta)$.

Choose a symmetric blurring function g, a **kernel** $k = g \star g$:

$$\begin{aligned} \mathrm{d}_k(\alpha, \beta) &= \| g \star \alpha - g \star \beta \|_{L^2}^2 \\ &= \langle \alpha - \beta \,|\, k \star (\alpha - \beta) \rangle \\ &= -2 \sum_{i,j} k(x_i, y_j)\, \alpha_i \, \beta_j + \cdots \end{aligned}$$

Kernel fidelities: the simplest formula for $d(\alpha, \beta)$



Blurred signal $g \star (\alpha - \beta)$.

Choose a symmetric blurring function g, a **kernel** $k = g \star g$:

$$\begin{aligned} \mathrm{d}_k(\alpha, \beta) &= \| g \star \alpha - g \star \beta \|_{L^2}^2 \\ &= \langle \alpha - \beta \,|\, k \star (\alpha - \beta) \rangle \\ &= -2 \sum_{i,j} k(x_i, y_j)\, \alpha_i\, \beta_j + \cdots \\ &= \langle \alpha - \beta \,|\, b^k - a^k \rangle \end{aligned}$$

with $a^k = -k \star \alpha$, $b^k = -k \star \beta$.

Kernel distances: distance fields computed through convolutions

Kernel distances, aka. **blurred** SSDs:

$$\text{choose} \qquad a(x) \;=\; -(k\star\alpha)(x) \;=\; -\sum_i \alpha_i\, k(x, x_i)$$

$$\text{and use} \qquad \tfrac{1}{2}\langle\, \alpha-\beta\,,\, b-a\,\rangle \;=\; \tfrac{1}{2}\langle\, \alpha-\beta\,,\, k\star(\alpha-\beta)\,\rangle.$$

Kernel distances: distance fields computed through convolutions

Kernel distances, aka. **blurred** SSDs:

$$\text{choose} \qquad a(x) \;=\; -(k \star \alpha)(x) \;=\; -\sum_i \alpha_i \, k(x, x_i)$$

$$\text{and use} \qquad \tfrac{1}{2}\langle\, \alpha - \beta \,,\; b - a \,\rangle \;=\; \tfrac{1}{2}\langle\, \alpha - \beta \,,\; k \star (\alpha - \beta) \,\rangle.$$

The **Energy Distance:** an underrated kernel, $k(x, y) \;=\; -\|x - y\|$.

$$a(x) \;=\; \sum_i \alpha_i \, \|x - x_i\| \qquad \text{instead of} \quad a(x) \;=\; \min_i \|x - x_i\|$$

$$b(x) \;=\; \sum_j \beta_j \, \|x - y_j\| \qquad \text{instead of} \quad b(x) \;=\; \min_j \|x - y_j\|.$$

Kernel distances: distance fields computed through convolutions

Kernel distances, aka. **blurred** SSDs:

$$\text{choose} \qquad a(x) \;=\; -(k \star \alpha)(x) \;=\; -\sum_i \alpha_i\, k(x, x_i)$$

$$\text{and use} \qquad \tfrac{1}{2}\langle\, \alpha - \beta \,,\, b - a \,\rangle \;=\; \tfrac{1}{2}\langle\, \alpha - \beta \,,\, k \star (\alpha - \beta)\,\rangle.$$

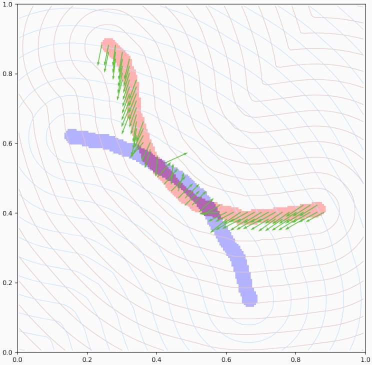
The **Energy Distance:** an underrated kernel, $k(x, y) \;=\; -\|x - y\|$.

$$a(x) \;=\; \sum_i \alpha_i\, \|x - x_i\| \qquad \text{instead of} \quad a(x) \;=\; \min_i \|x - x_i\|$$

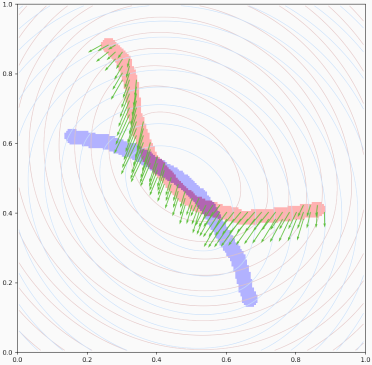
$$b(x) \;=\; \sum_j \beta_j\, \|x - y_j\| \qquad \text{instead of} \quad b(x) \;=\; \min_j \|x - y_j\|.$$

$$\begin{aligned}\text{Loss}(\alpha, \beta) \;=\; & \sum_i \sum_j \alpha_i \beta_j\, \|x_i - y_j\| \\ & - \tfrac{1}{2} \sum_i \sum_j \alpha_i \alpha_j\, \|x_i - x_j\| \;-\; \tfrac{1}{2} \sum_i \sum_j \beta_i \beta_j\, \|y_i - y_j\|\end{aligned}$$

The Hausdorff distance is local, the Energy Distance is global

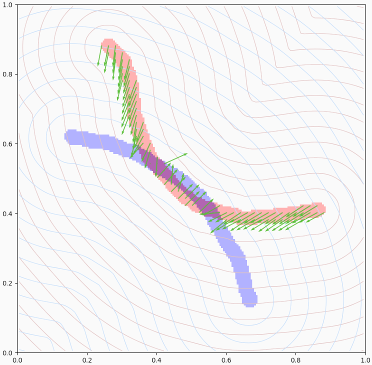


Hausdorff, min

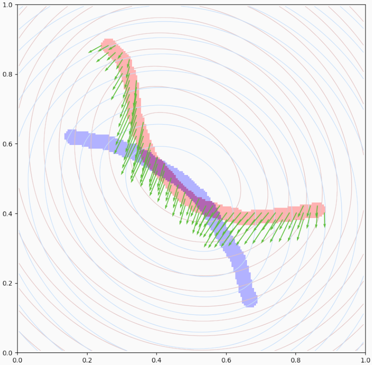


Kernel, $\sum$

The Hausdorff distance is local, the Energy Distance is global



Hausdorff, min

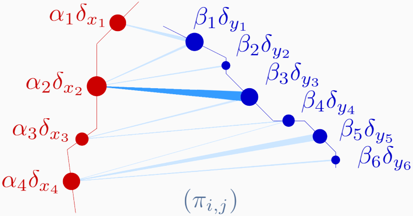


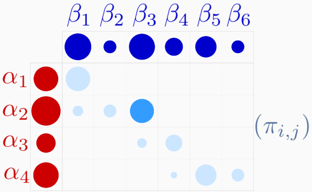
Kernel, $\sum$

$\Longrightarrow$ Can we get the best of both worlds?

An idea from Optimal Transport theory: The SoftAssign algorithm

Introducing the Optimal Transport problem





Minimize over N-by-M matrices (transport plans) π :

$$\text{OT}(\alpha, \beta) \;=\; \min_{\pi} \underbrace{\sum_{i,j} \pi_{i,j} \cdot |x_i - y_j|^2}_{\text{transport cost}}$$

subject to $\quad \pi_{i,j} \;\geqslant\; 0,$

$$\sum_j \pi_{i,j} \;=\; \alpha_i, \quad \sum_i \pi_{i,j} \;=\; \beta_j.$$

Kantorovitch's dual formulation

With $\mathrm{C}(x_i, y_j) = \|x_i - y_j\|^p$,

$$\mathrm{OT}(\alpha, \beta) = \min_{\pi} \langle \pi , \mathrm{C} \rangle \qquad \longrightarrow \text{Assignment}$$

$$\text{s.t. } \pi \geqslant 0, \qquad \pi \mathbf{1} = \alpha, \qquad \pi^{\intercal} \mathbf{1} = \beta$$

Kantorovitch's dual formulation

With $C(x_i, y_j) = \|x_i - y_j\|^p$,

$$\mathrm{OT}(\alpha, \beta) = \min_{\pi} \langle \pi , C \rangle \qquad \longrightarrow \text{Assignment}$$

$$\text{s.t. } \pi \geqslant 0, \qquad \pi \mathbf{1} = \alpha, \qquad \pi^{\top} \mathbf{1} = \beta$$

$$= \max_{f, g} \langle \alpha , f \rangle + \langle \beta , g \rangle \qquad \longrightarrow \text{FedEx}$$

$$\text{s.t. } \qquad f(x_i) + g(y_j) \leqslant C(x_i, y_j),$$

Kantorovitch's dual formulation

With $C(x_i, y_j) = \|x_i - y_j\|^p$,

$$\mathrm{OT}(\alpha, \beta) = \min_{\pi} \langle \pi , C \rangle \qquad \longrightarrow \text{Assignment}$$

$$\text{s.t. } \pi \geqslant 0, \qquad \pi \mathbf{1} = \alpha, \qquad \pi^{\top} \mathbf{1} = \beta$$

$$= \max_{f, g} \langle \alpha , f \rangle + \langle \beta , g \rangle \qquad \longrightarrow \text{FedEx}$$

$$\text{s.t. } \qquad f(x_i) + g(y_j) \leqslant C(x_i, y_j),$$

$\Longrightarrow$ **Combinatorial** problem on the simplex

Kantorovitch's dual formulation

With $C(x_i, y_j) = \|x_i - y_j\|^p$,

$$\mathrm{OT}(\alpha, \beta) = \min_{\pi} \langle \pi , C \rangle \qquad \longrightarrow \text{Assignment}$$

$$\text{s.t. } \pi \geqslant 0, \qquad \pi \mathbf{1} = \alpha, \qquad \pi^\top \mathbf{1} = \beta$$

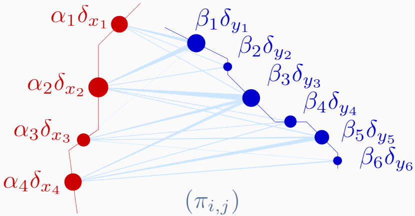
$$= \max_{f, g} \langle \alpha , f \rangle + \langle \beta , g \rangle \qquad \longrightarrow \text{FedEx}$$

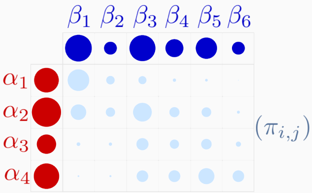
$$\text{s.t.} \qquad f(x_i) + g(y_j) \leqslant C(x_i, y_j),$$

$\Longrightarrow$ **Combinatorial** problem on the simplex

$\Longrightarrow$ Hungarian method in $\mathbf{O}(\mathbf{N}^3)$.

Entropic regularization: introducing Schrödinger's problem





For $\varepsilon > 0$:

$$\mathrm{OT}_\varepsilon(\alpha, \beta) \;=\; \min_\pi \underbrace{\sum_{i,j} \pi_{i,j} \cdot |x_i - y_j|^2}_{\text{transport cost}}$$

$$+ \;\varepsilon \underbrace{\sum_{i,j} \pi_{i,j} \cdot \log \frac{\pi_{i,j}}{\alpha_i \, \beta_j}}_{\text{entropic barrier}}$$

subject to

$$\sum_j \pi_{i,j} \;=\; \alpha_i, \quad \sum_i \pi_{i,j} \;=\; \beta_j.$$

Fenchel-Rockafellar to the rescue

$$\mathrm{OT}_\varepsilon(\alpha, \beta) = \min_{\pi} \langle \pi , \mathrm{C} \rangle + \varepsilon \, \mathrm{KL}(\pi , \alpha \otimes \beta) \quad \longrightarrow \text{ Fuzzy assignment}$$

$$\text{s.t.} \qquad \pi \, \mathbf{1} = \alpha, \qquad \pi^\intercal \mathbf{1} = \beta$$

Fenchel-Rockafellar to the rescue

$$\mathrm{OT}_\varepsilon(\alpha,\beta) \;=\; \min_\pi \,\langle\, \pi\,,\, \mathrm{C}\,\rangle \;+\; \varepsilon\,\mathrm{KL}(\,\pi\,,\,\alpha\otimes\beta\,) \quad\longrightarrow \text{Fuzzy assignment}$$

$$\text{s.t.} \qquad \pi\,\mathbf{1} \;=\; \alpha, \qquad \pi^\intercal\mathbf{1} \;=\; \beta$$

$$= \max_{f,g} \,\langle\, \alpha\,,\, f \rangle \;+\; \langle\, \beta\,,\, g\,\rangle \quad\longrightarrow \text{Cheeky FedEx}$$

$$-\; \underbrace{\varepsilon\,\langle\, \alpha\otimes\beta\,,\, e^{(f\oplus g-\mathrm{C})/\varepsilon} - 1\,\rangle}_{\text{soft constraint } f\oplus g\,\leqslant\,\mathrm{C}}$$

Fenchel-Rockafellar to the rescue

$$\begin{aligned} \mathrm{OT}_\varepsilon(\alpha,\beta) \;&=\; \min_{\pi} \langle \pi , \mathrm{C} \rangle + \varepsilon\, \mathrm{KL}(\pi , \alpha\otimes\beta) && \longrightarrow \text{Fuzzy assignment} \\ \text{s.t.} \quad & \pi\mathbf{1} = \alpha, \qquad \pi^\intercal\mathbf{1} = \beta \\ &= \max_{f,g} \langle \alpha , f\rangle + \langle \beta , g\rangle && \longrightarrow \text{Cheeky FedEx} \\ & \quad - \underbrace{\varepsilon \langle \alpha\otimes\beta , e^{(f\oplus g - \mathrm{C})/\varepsilon} - 1 \rangle}_{\text{soft constraint } f\oplus g \leqslant \mathrm{C}} \end{aligned}$$

$\Longrightarrow$ **Strictly convex** problem on the simplex

Fenchel-Rockafellar to the rescue

$$\mathrm{OT}_\varepsilon(\alpha, \beta) = \min_{\pi} \langle \pi, \mathrm{C} \rangle + \varepsilon\, \mathrm{KL}(\pi, \alpha \otimes \beta) \quad \longrightarrow \text{Fuzzy assignment}$$

$$\text{s.t.} \quad \pi \mathbf{1} = \alpha, \qquad \pi^\intercal \mathbf{1} = \beta$$

$$= \max_{f, g} \langle \alpha, f \rangle + \langle \beta, g \rangle \quad \longrightarrow \text{Cheeky FedEx}$$

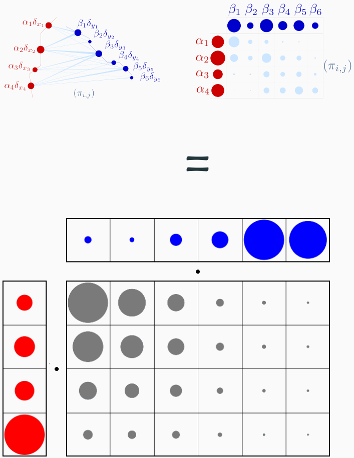
$$- \underbrace{\varepsilon \langle \alpha \otimes \beta, e^{(f \oplus g - \mathrm{C})/\varepsilon} - 1 \rangle}_{\text{soft constraint } f \oplus g \leqslant \mathrm{C}}$$

$\Longrightarrow$ **Strictly convex** problem on the simplex

$$\text{At the optimum,} \quad \pi = e^{(f \oplus g - \mathrm{C})/\varepsilon} \cdot \alpha \otimes \beta$$

$$\text{i.e.} \quad \pi_{i,j} = \alpha_i\, e^{f_i/\varepsilon}\, e^{-\mathrm{C}(x_i, y_j)/\varepsilon}\, e^{g_j/\varepsilon}\, \beta_j.$$

Textbook interpretation: balancing of a kernel matrix



$$\pi_{i,j} \;=\; \Delta(U\alpha) \cdot \mathrm{K}_{x,y} \cdot \Delta(V\beta)$$

with

- a kernel function k

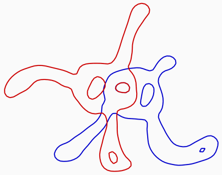
$$k(x_i - y_j) \;=\; e^{-\mathrm{C}(x_i,y_j)/\varepsilon}.$$

- $U = e^{f/\varepsilon}$ and $V = e^{g/\varepsilon}$, positive weights on $\{x_i\}$ and $\{y_j\}$.

$\rightarrow$ Enforce the **constraints**

$$\pi\,\mathbf{1} \;=\; \alpha, \qquad \pi^{\top}\mathbf{1} \;=\; \beta$$

Enforcing $\pi \mathbb{1} = \alpha$ and $\pi^\top \mathbb{1} = \beta$ alternatively



Source and target.

Sinkhorn Iterative Algorithm

Input : source $\alpha = \sum_i \alpha_i \delta_{x_i}$

target $\beta = \sum_j \beta_j \delta_{y_j}$

Parameter : $k : x \mapsto e^{-|x|^2/\varepsilon}$

1: $U \leftarrow ones(size(\alpha))$

2: $V \leftarrow ones(size(\beta))$

3: **while** updates $>$ tol **do**

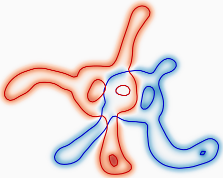
4: $U \leftarrow \mathbb{1} \;./\; K \cdot (V\beta)$

5: $V \leftarrow \mathbb{1} \;./\; K^\top \cdot (U\alpha)$

6: **return** $\varepsilon \left(\langle \alpha, \log(U) \rangle + \langle \beta, \log(V) \rangle \right)$

Output : fidelity $\mathrm{OT}_\varepsilon(\alpha, \beta)$

Enforcing $\pi 1 = \alpha$ and $\pi^\top 1 = \beta$ alternatively



Seen by the kernel k.

Sinkhorn Iterative Algorithm

Input : source $\alpha = \sum_i \alpha_i \delta_{x_i}$

target $\beta = \sum_j \beta_j \delta_{y_j}$

Parameter : $k : x \mapsto e^{-|x|^2/\varepsilon}$

$U \leftarrow ones(size(\alpha))$

$V \leftarrow ones(size(\beta))$

while updates > tol **do**

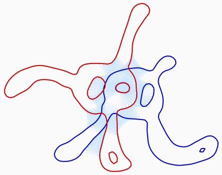
$\quad U \leftarrow 1 \;./\; K \cdot (V\beta)$

$\quad V \leftarrow 1 \;./\; K^\top \cdot (U\alpha)$

return $\varepsilon\,(\,\langle\,\alpha, \log(U)\,\rangle + \langle\,\beta, \log(V)\,\rangle\,)$

Output : fidelity $\mathrm{OT}_\varepsilon(\alpha, \beta)$

Enforcing $\pi 1 = \alpha$ and $\pi^\top 1 = \beta$ alternatively



Starting estimate.

Sinkhorn Iterative Algorithm

Input : source $\alpha = \sum_i \alpha_i \delta_{x_i}$

target $\beta = \sum_j \beta_j \delta_{y_j}$

Parameter : $k : x \mapsto e^{-|x|^2/\varepsilon}$

1: $U \leftarrow ones(size(\alpha))$

2: $V \leftarrow ones(size(\beta))$

3: **while** updates $>$ tol **do**

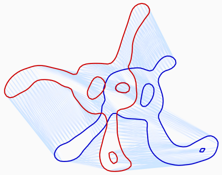
4: $\quad U \leftarrow 1 \;./\; \mathrm{K} \cdot (V\beta)$

5: $\quad V \leftarrow 1 \;./\; \mathrm{K}^\top \cdot (U\alpha)$

6: **return** $\varepsilon \left(\langle \alpha, \log(U) \rangle + \langle \beta, \log(V) \rangle \right)$

Output : fidelity $\mathrm{OT}_\varepsilon(\alpha, \beta)$

Enforcing $\pi 1 = \alpha$ and $\pi^{\intercal} 1 = \beta$ alternatively



Computing the OT plan.

Sinkhorn Iterative Algorithm

Input : source $\alpha = \sum_i \alpha_i \delta_{x_i}$

target $\beta = \sum_j \beta_j \delta_{y_j}$

Parameter : $k : x \mapsto e^{-|x|^2/\varepsilon}$

1: $U \leftarrow ones(size(\alpha))$

2: $V \leftarrow ones(size(\beta))$

3: **while** updates $>$ tol **do**

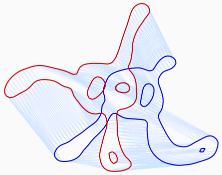
4: $\quad U \leftarrow \mathbf{1} \;./\; \mathrm{K} \cdot (V\beta)$

5: $\quad V \leftarrow \mathbf{1} \;./\; \mathrm{K}^{\intercal} \cdot (U\alpha)$

6: **return** $\varepsilon \left(\langle \alpha, \log(U) \rangle + \langle \beta, \log(V) \rangle \right)$

Output : fidelity $\mathrm{OT}_\varepsilon(\alpha, \beta)$

Enforcing $\pi \mathbb{1} = \alpha$ and $\pi^\top \mathbb{1} = \beta$ alternatively



Sinkhorn Iteration 250

Computing the OT plan.

Sinkhorn Iterative Algorithm

Input : source $\alpha = \sum_i \alpha_i \delta_{x_i}$

target $\beta = \sum_j \beta_j \delta_{y_j}$

Parameter : $k : x \mapsto e^{-|x|^2/\varepsilon}$

1: $U \leftarrow ones(size(\alpha))$

2: $V \leftarrow ones(size(\beta))$

3: **while** updates > tol **do**

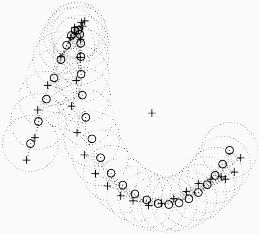
4: $\quad U \leftarrow \mathbb{1} \,./\, \mathrm{K} \cdot (V\beta)$

5: $\quad V \leftarrow \mathbb{1} \,./\, \mathrm{K}^\top \cdot (U\alpha)$

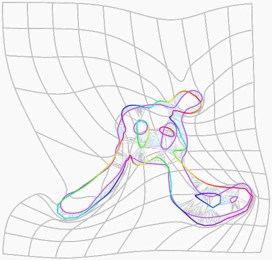
6: **return** $\varepsilon \left(\langle \alpha, \log(U) \rangle + \langle \beta, \log(V) \rangle \right)$

Output : fidelity $\mathrm{OT}_\varepsilon(\alpha, \beta)$

Robust Point Matching, 1998-2017



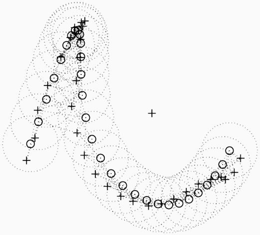
$\simeq$



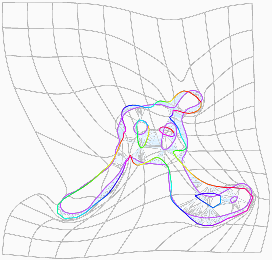
TPS-RPM algorithm,
Chui and Rangarajan, CVPR **2000**

Optimal Transport for diffeomorphic registration, Feydy et al., MICCAI **2017**

Robust Point Matching, 1998-2017



$\simeq$



TPS-RPM algorithm,
Chui and Rangarajan, CVPR **2000**

Optimal Transport for diffeomorphic registration, Feydy et al., MICCAI **2017**

$\Longrightarrow$ We've added weights, orientations, convergence analysis...
But shouldn't we go a bit **further**?

It’s 2018 now:
What’s new?

Fact 1 : Sinkhorn is best implemented in the log-domain

Unfortunately,

$$k(x_i, y_j) \;\simeq\; 0 \qquad\qquad \text{if} \quad \varepsilon \quad \text{is too small.}$$

Fact 1 : Sinkhorn is best implemented in the log-domain

Unfortunately,

$$k(x_i, y_j) \simeq 0 \qquad \text{if} \quad \varepsilon \quad \text{is too small.}$$

$$\begin{aligned} \mathrm{OT}_\varepsilon(\alpha, \beta) \;=\; & \max_{f,g} \langle \alpha , f \rangle + \langle \beta , g \rangle \qquad \longrightarrow \text{Cheeky FedEx} \\ & - \underbrace{\varepsilon \langle \alpha \otimes \beta , e^{(f \oplus g - \mathrm{C})/\varepsilon} - 1 \rangle}_{\text{soft constraint } f \oplus g \leqslant \mathrm{C}} \end{aligned}$$

Fact 1 : Sinkhorn is best implemented in the log-domain

Unfortunately,

$$k(x_i, y_j) \simeq 0 \qquad\qquad \text{if } \ \varepsilon \ \text{ is too small.}$$

$$\begin{aligned}\mathrm{OT}_\varepsilon(\alpha, \beta) \;=\; \max_{f,g}\, \langle\, \alpha\,, f\,\rangle \,+\, \langle\, \beta\,, g\,\rangle \qquad\qquad \longrightarrow \text{ Cheeky FedEx}\\ -\; \underbrace{\varepsilon\,\langle\, \alpha\otimes\beta\,,\; e^{(f\oplus g - \mathrm{C})/\varepsilon} - 1\,\rangle}_{\text{soft constraint } f\oplus g\,\leqslant\,\mathrm{C}}\end{aligned}$$

Equivalent to the constraints on π, the optimality conditions read:

$$f(x_i) \;=\; -\varepsilon\,\log \sum_j \beta_j\, \exp \tfrac{1}{\varepsilon}\big(g(y_j) - \mathrm{C}(x_i, y_j)\big),$$

$$g(y_j) \;=\; -\varepsilon\,\log \sum_i \alpha_i\, \exp \tfrac{1}{\varepsilon}\big(f(x_i) - \mathrm{C}(x_i, y_j)\big).$$

The SoftMin interpolates between a minimum and a sum

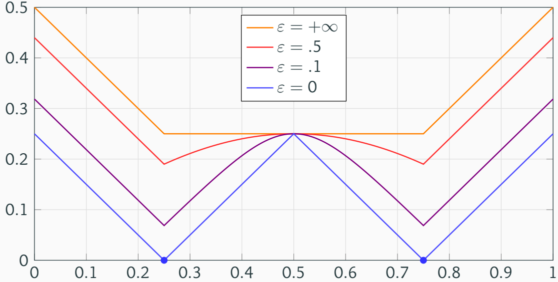
$$\log\Big(e^c \, + \, e^d \Big) \;\; = \;\; \max(c,d) \, + \, \log\big(\underbrace{e^{c-\max(c,d)} \, + \, e^{d-\max(c,d)}}_{\in [1,2]} \big)$$

The SoftMin interpolates between a minimum and a sum

$$\log\Big(e^{c} + e^{d} \Big) = \max(c,d) + \log\big(\underbrace{e^{c-\max(c,d)} + e^{d-\max(c,d)}}_{\in[1,2]} \big)$$

Building on this, for a regularization parameter $\varepsilon > 0$, we define

$$b^{\varepsilon}(x) = \min_{y\sim\beta}{}_{\varepsilon} \|x-y\| = -\varepsilon \log \sum_{j=1}^{M} \beta_j \exp\big(-\tfrac{1}{\varepsilon}\|x-y_j\|\big)$$



$b^{\varepsilon}(x)$, with $\beta = \frac{1}{2}\delta_{.25} + \frac{1}{2}\delta_{.75}$

A new reduction on the distance matrix

$$
\begin{array}{cc}
 & \begin{array}{cccc} \beta_1 & \beta_2 & \cdots & \beta_M \end{array} \\
\begin{array}{c} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{array} &
\begin{pmatrix}
\|x_1 - y_1\| & \|x_1 - y_2\| & \cdots & \|x_1 - y_M\| \\
\|x_2 - y_1\| & \|x_2 - y_2\| & \cdots & \|x_2 - y_M\| \\
\vdots & \vdots & \ddots & \vdots \\
\|x_N - y_1\| & \|x_N - y_2\| & \cdots & \|x_N - y_M\|
\end{pmatrix}
\end{array}
$$

A new reduction on the distance matrix

$$\begin{array}{cc} & \begin{array}{cccc} \beta_1 & \beta_2 & \cdots & \beta_M \end{array} \\ \begin{array}{c} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{array} & \begin{pmatrix} \|x_1 - y_1\| & \|x_1 - y_2\| & \cdots & \|x_1 - y_M\| \\ \|x_2 - y_1\| & \|x_2 - y_2\| & \cdots & \|x_2 - y_M\| \\ \vdots & \vdots & \ddots & \vdots \\ \|x_N - y_1\| & \|x_N - y_2\| & \cdots & \|x_N - y_M\| \end{pmatrix} \end{array}$$

A new reduction on the distance matrix

$$
\begin{array}{cc}
 & \begin{array}{cccc} \beta_1 & \beta_2 & \cdots & \beta_M \end{array} \\
\begin{array}{c} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{array} &
\begin{pmatrix}
\|x_1 - y_1\| & \|x_1 - y_2\| & \cdots & \|x_1 - y_M\| \\
\|x_2 - y_1\| & \|x_2 - y_2\| & \cdots & \|x_2 - y_M\| \\
\vdots & \vdots & \ddots & \vdots \\
\|x_N - y_1\| & \|x_N - y_2\| & \cdots & \|x_N - y_M\|
\end{pmatrix}
\begin{array}{c} \to \quad \sum_j \beta_j \|x_1 - y_j\| \\ \to \quad \sum_j \beta_j \|x_2 - y_j\| \\ \vdots \\ \to \quad \sum_j \beta_j \|x_N - y_j\| \end{array}
\end{array}
$$

Energy Distance : $\sum_j \beta_j \|x_i - y_j\| = b_k(x_i)$

A new reduction on the distance matrix

$$
\begin{array}{cc}
 & \begin{array}{cccc} \beta_1 & \beta_2 & \cdots & \beta_M \end{array} \\
\begin{array}{c} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{array} &
\begin{pmatrix}
\|x_1 - y_1\| & \|x_1 - y_2\| & \cdots & \|x_1 - y_M\| \\
\|x_2 - y_1\| & \|x_2 - y_2\| & \cdots & \|x_2 - y_M\| \\
\vdots & \vdots & \ddots & \vdots \\
\|x_N - y_1\| & \|x_N - y_2\| & \cdots & \|x_N - y_M\|
\end{pmatrix}
\begin{array}{l} \to \quad \min_j \|x_1 - y_j\| \\ \to \quad \min_j \|x_2 - y_j\| \\ \vdots \\ \to \quad \min_j \|x_N - y_j\| \end{array}
\end{array}
$$

Energy Distance : $\sum_j \beta_j \|x_i - y_j\| \quad = \quad b_k(x_i)$

Hausdorff Distance : $\min_j \|x_i - y_j\| \quad = \quad \mathrm{d}(x_i, \mathrm{supp}(\beta))$

A new reduction on the distance matrix

$$
\begin{array}{cc}
 & \begin{array}{cccc} \beta_1 & \beta_2 & \cdots & \beta_M \end{array} \\
\begin{array}{c} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{array} &
\begin{pmatrix}
\|x_1 - y_1\| & \|x_1 - y_2\| & \cdots & \|x_1 - y_M\| \\
\|x_2 - y_1\| & \|x_2 - y_2\| & \cdots & \|x_2 - y_M\| \\
\vdots & \vdots & \ddots & \vdots \\
\|x_N - y_1\| & \|x_N - y_2\| & \cdots & \|x_N - y_M\|
\end{pmatrix}
\begin{array}{l}
\to \quad \min_{\varepsilon, y\sim\beta} \|x_1 - y\| \\
\to \quad \min_{\varepsilon, y\sim\beta} \|x_2 - y\| \\
\vdots \\
\to \quad \min_{\varepsilon, y\sim\beta} \|x_N - y\|
\end{array}
\end{array}
$$

$$
\begin{array}{lcccl}
\text{Energy Distance} & : & \sum_j \beta_j \|x_i - y_j\| & = & b_k(x_i) \\
\varepsilon\text{-SoftMin} & : & \min_{\varepsilon, y\sim\beta} \|x_i - y\| & = & b_\varepsilon(x_i) \simeq f(x_i) \\
\text{Hausdorff Distance} & : & \min_j \|x_i - y_j\| & = & \mathrm{d}(x_i, \mathrm{supp}(\beta))
\end{array}
$$

Optimal Transport = Hausdorff + mass spreading constraint

The optimality conditions read:

$$f(x_i) \;=\; b(x) \;=\; -\varepsilon\,\log \sum_j \beta_j \,\exp \tfrac{1}{\varepsilon}\big[\,g(y_j) - \mathrm{C}(x_i,y_j)\,\big],$$

$$g(y_j) \;=\; a(y) \;=\; -\varepsilon\,\log \sum_i \alpha_i \,\exp \tfrac{1}{\varepsilon}\big[\,f(x_i) - \mathrm{C}(x_i,y_j)\,\big].$$

Optimal Transport = Hausdorff + mass spreading constraint

The optimality conditions read:

$$f(x_i) \;=\; b(x) \;=\; \min_{\varepsilon,\, y\sim\beta} \big[\, \mathrm{C}(x,y) \;-\; a(y) \,\big] \qquad ,$$

$$g(y_j) \;=\; a(y) \;=\; \min_{\varepsilon,\, x\sim\alpha} \big[\, \mathrm{C}(x,y) \;-\; b(x) \,\big] \qquad .$$

Optimal Transport = Hausdorff + mass spreading constraint

The optimality conditions read:

$$f(x_i) \;=\; b(x) \;=\; \min_{y\sim\beta}{}_{\varepsilon} \big[\, \mathrm{C}(x,y) \;-\; a(y) \,\big] \qquad ,$$

$$g(y_j) \;=\; a(y) \;=\; \min_{x\sim\alpha}{}_{\varepsilon} \big[\, \mathrm{C}(x,y) \;-\; b(x) \,\big] \qquad .$$

Final cost:

$$\mathrm{OT}_\varepsilon(\alpha,\beta) \;=\; \langle\, \alpha \,,\, f \,\rangle \,+\, \langle\, \beta \,,\, g \,\rangle \;=\; \langle\, \alpha \,,\, b \,\rangle \,+\, \langle\, \beta \,,\, a \,\rangle.$$

Optimal Transport = Hausdorff + mass spreading constraint

The optimality conditions read:

$$f(x_i) \;=\; b(x) \;=\; \min_{y\sim\beta}{}_{\varepsilon}\,\big[\,\mathrm{C}(x,y)\;-\;a(y)\,\big] \qquad ,$$

$$g(y_j) \;=\; a(y) \;=\; \min_{x\sim\alpha}{}_{\varepsilon}\,\big[\,\mathrm{C}(x,y)\;-\;b(x)\,\big] \qquad .$$

Final cost:

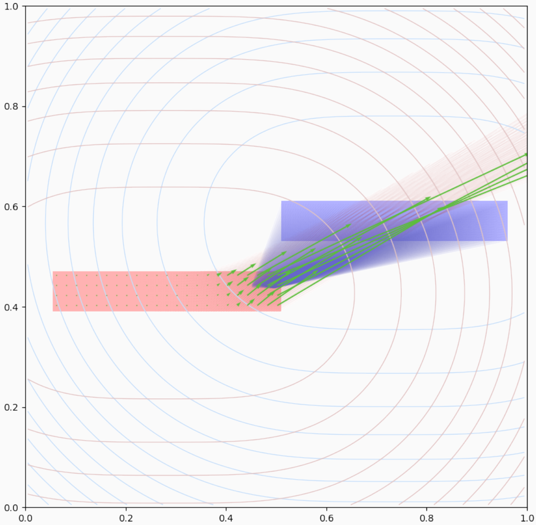
$$\mathrm{OT}_\varepsilon(\alpha,\beta) \;=\; \langle\,\alpha\,,\,f\,\rangle + \langle\,\beta\,,\,g\,\rangle \;=\; \langle\,\alpha\,,\,b\,\rangle + \langle\,\beta\,,\,a\,\rangle.$$

Discrete, computational OT [Cuturi, 2013, Peyré and Cuturi, 2018]:

Start from an ε-smoothed **Hausdorff** distance, but let the influence fields a and b **interact** with each other.

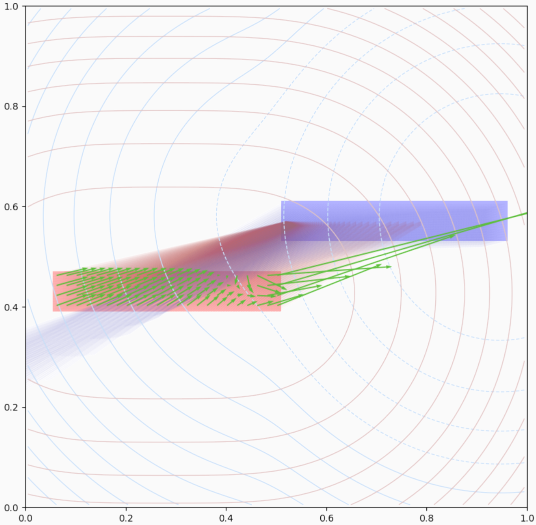
Enforce a **mass spreading** constraint on the spring system: all of α should be linked to all of β.

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



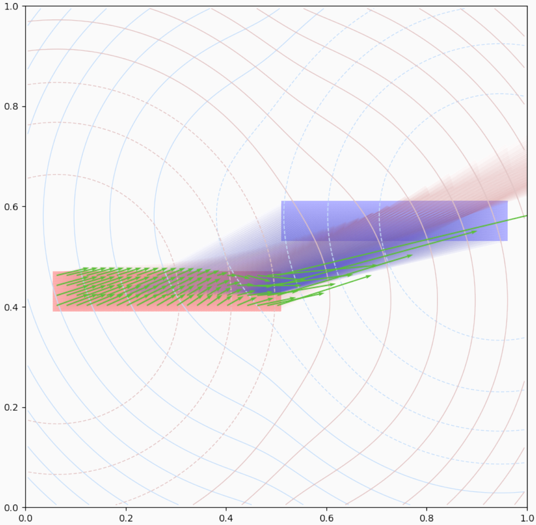
Iteration 1/2

The Sinkhorn algorithm, in practice; with $\|x - y\|^2$ and $\sqrt{\varepsilon} = .1$



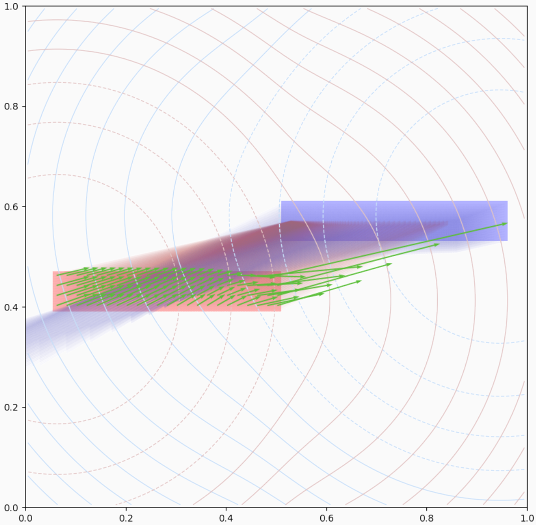
Iteration 1

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



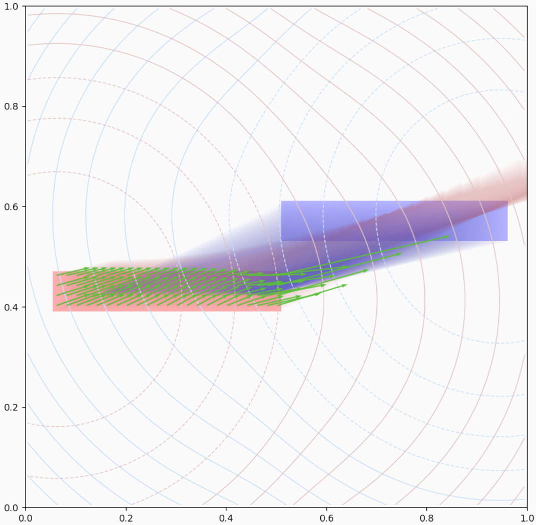
Iteration 1 ½

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



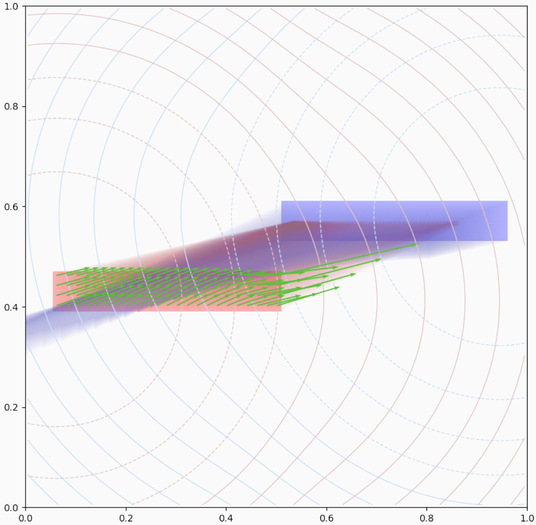
Iteration 2

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



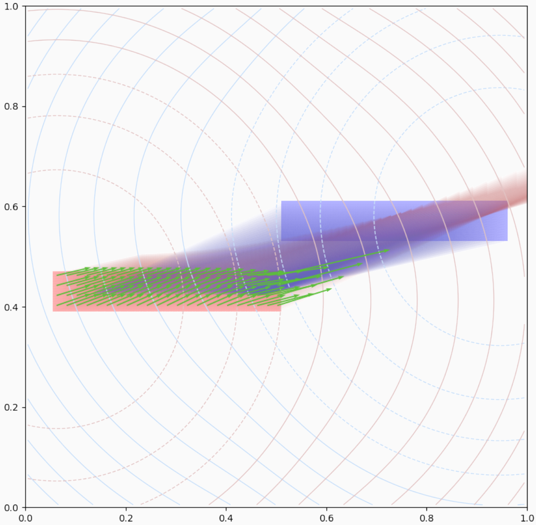
Iteration 2 ½

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



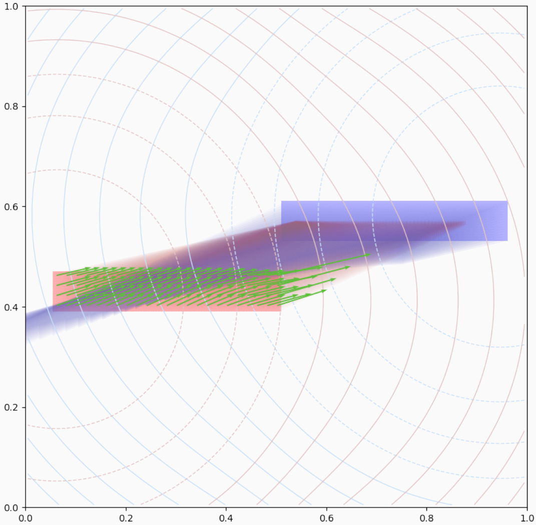
Iteration 3

The Sinkhorn algorithm, in practice; with $\|x - y\|^2$ and $\sqrt{\varepsilon} = .1$



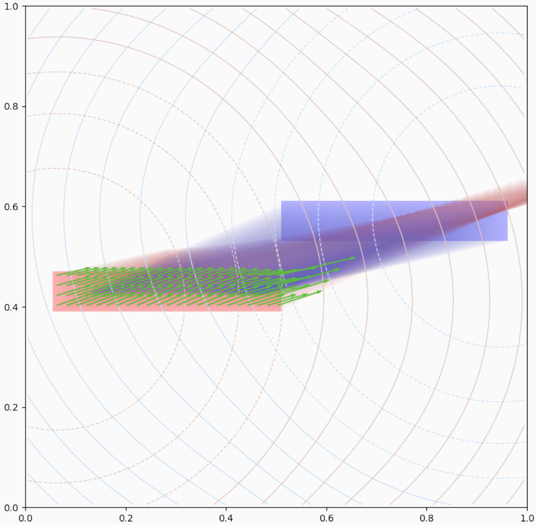
Iteration 3 ½

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



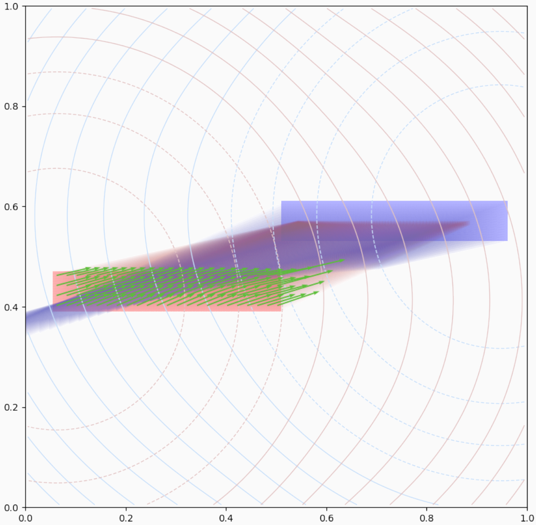
Iteration 4

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



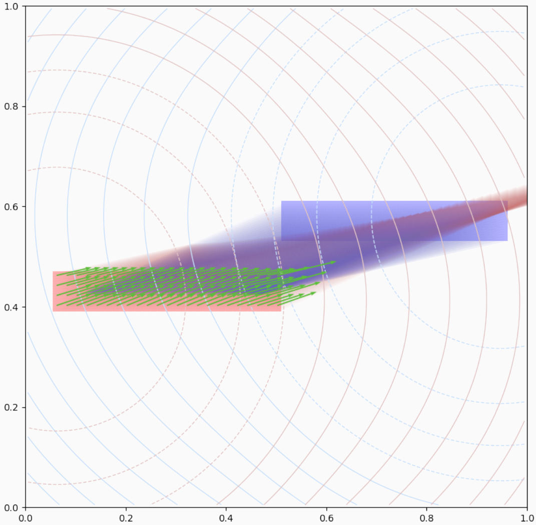
Iteration 4 ½

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



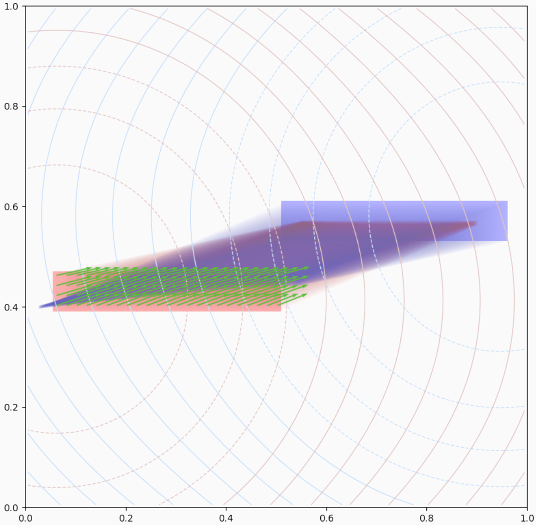
Iteration 5

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



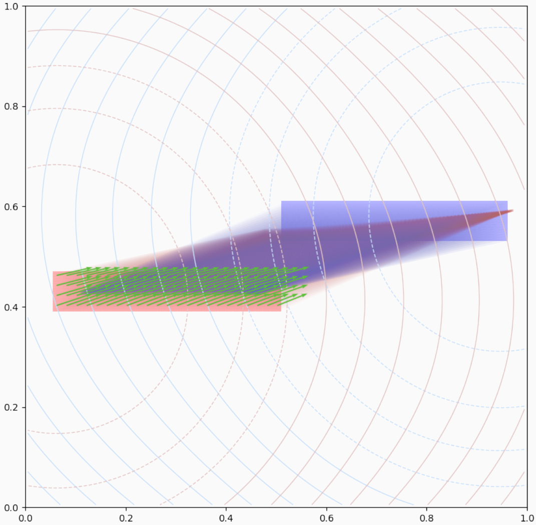
Iteration 5 ½

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



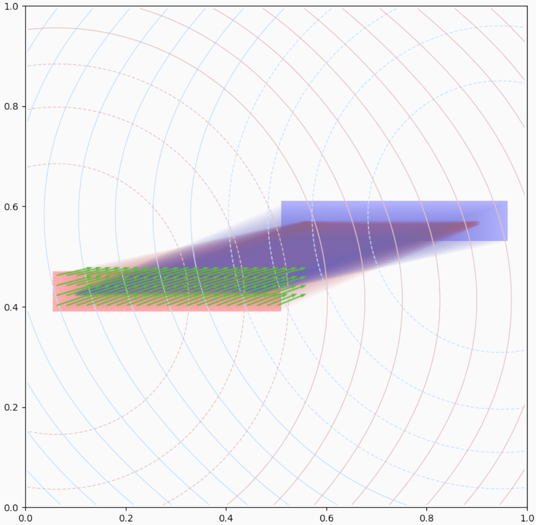
Iteration 10

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



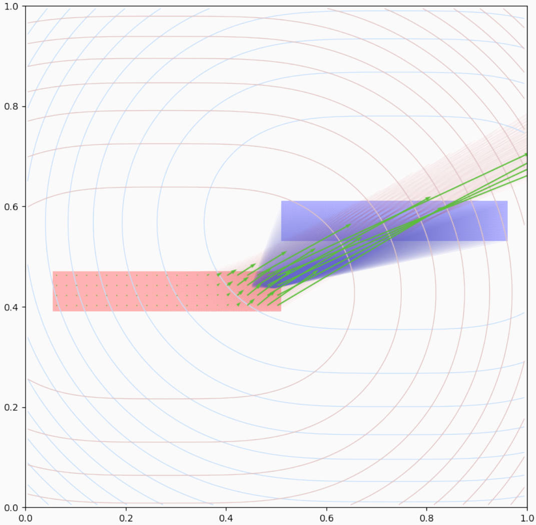
Iteration 10 ½

The Sinkhorn algorithm, in practice; with $\|x - y\|^2$ and $\sqrt{\varepsilon} = .1$



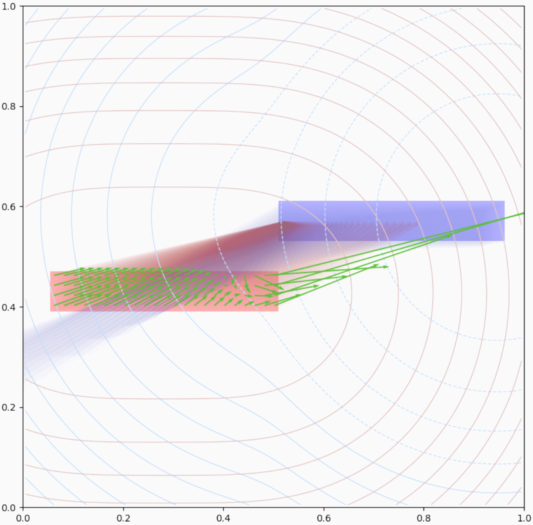
Iteration 20

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



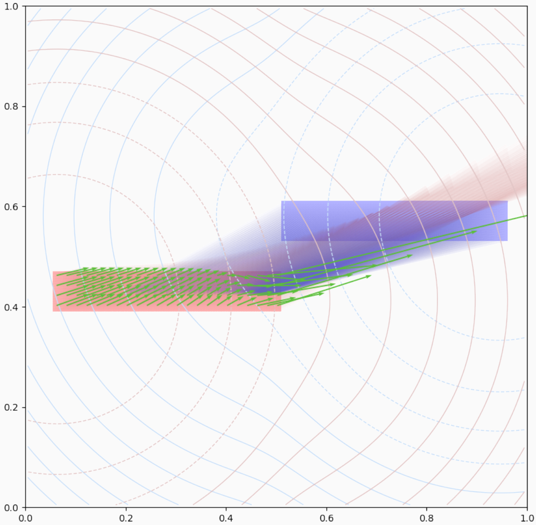
Iteration 1/2

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



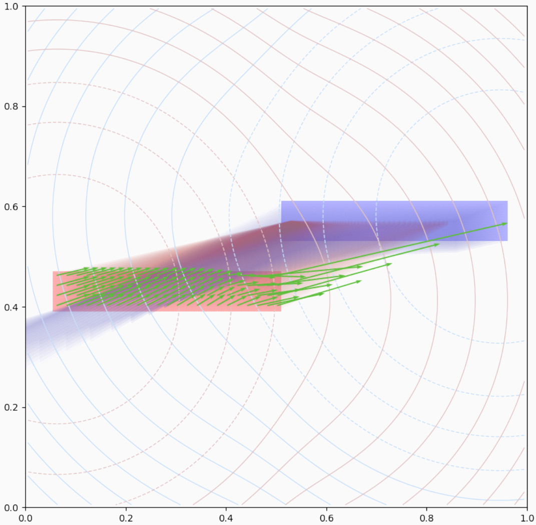
Iteration 1

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



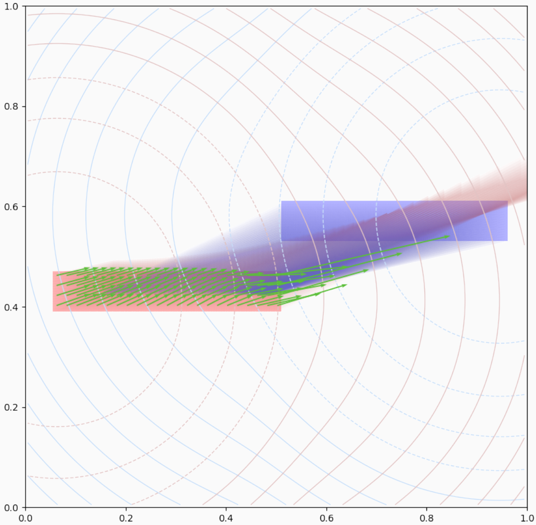
Iteration 1 ½

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



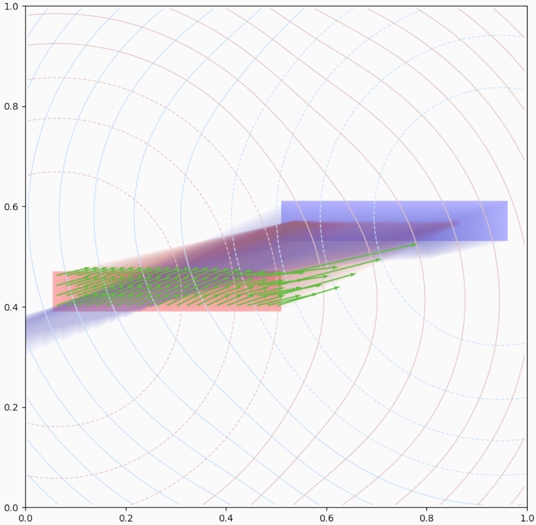
Iteration 2

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



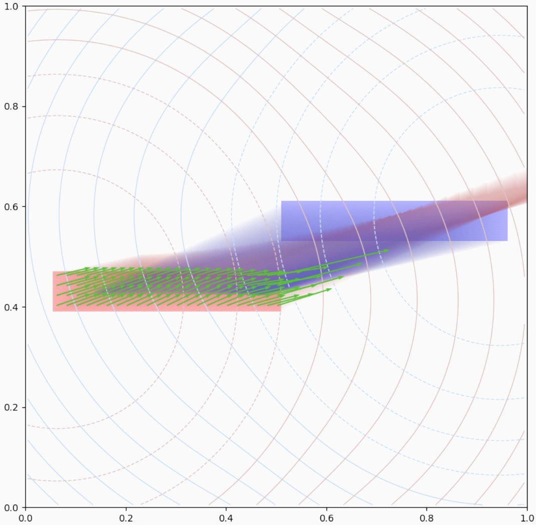
Iteration 2 ½

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



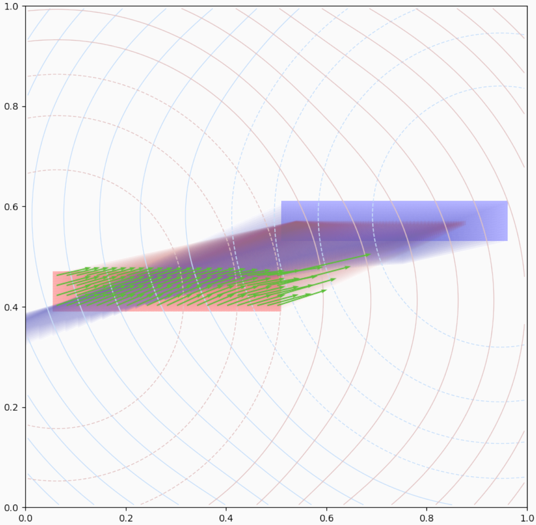
Iteration 3

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



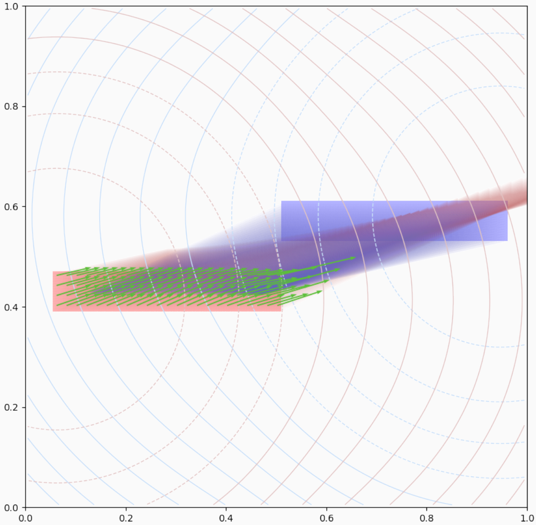
Iteration 3 ½

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



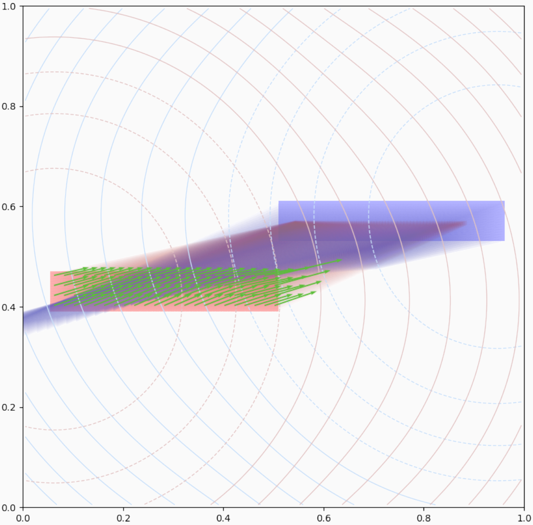
Iteration 4

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



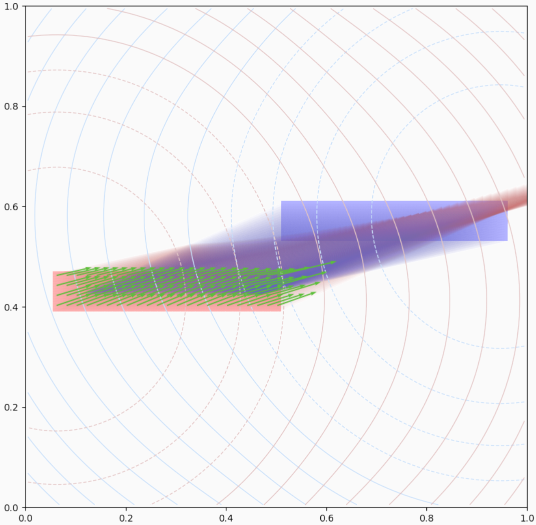
Iteration 4 ½

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



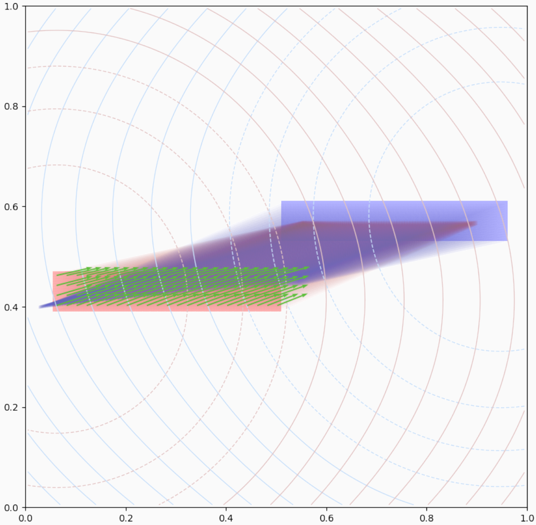
Iteration 5

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



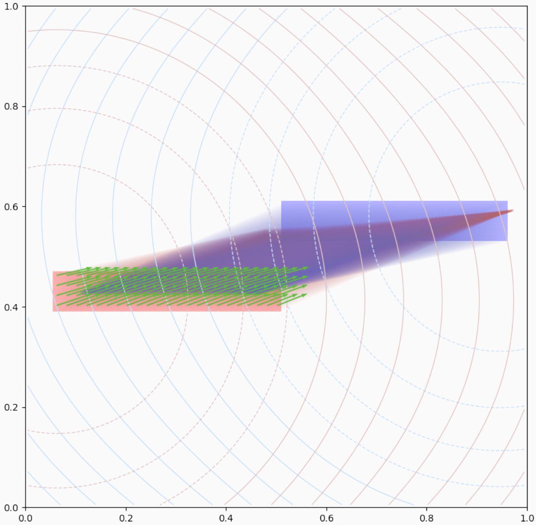
Iteration 5 ½

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



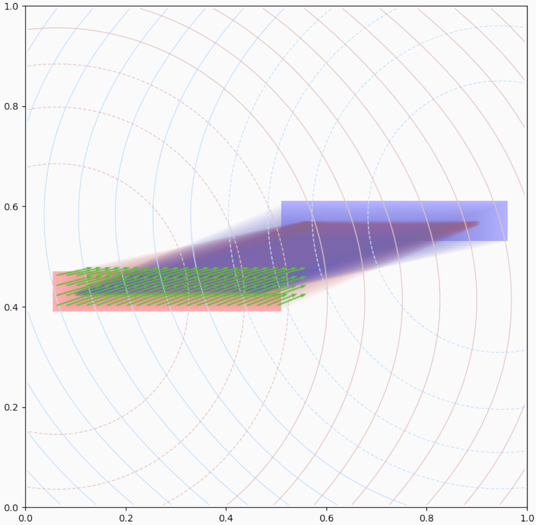
Iteration 10

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



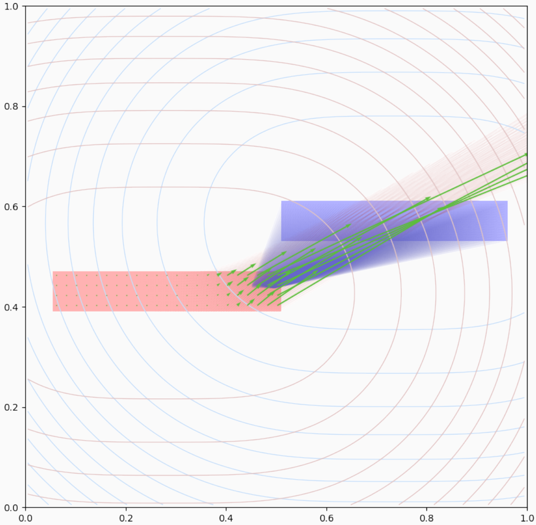
Iteration 10 ½

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



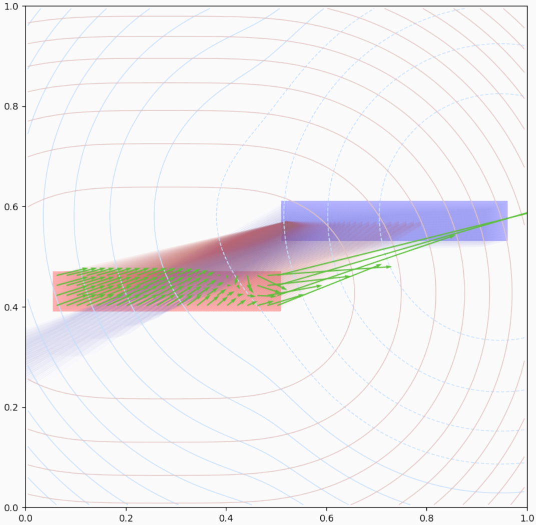
Iteration 20

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



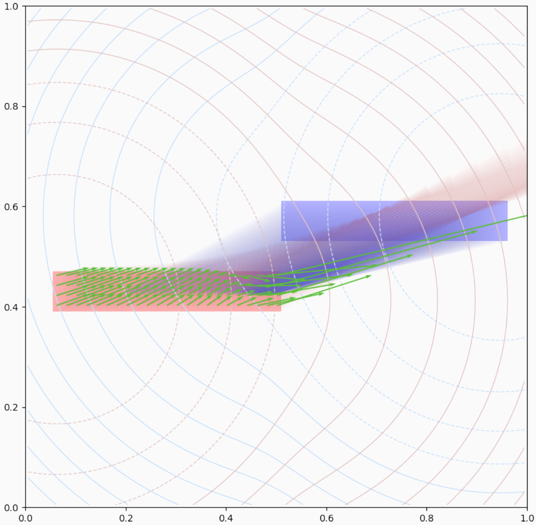
Iteration 1/2

The Sinkhorn algorithm, in practice; with $\|x - y\|^2$ and $\sqrt{\varepsilon} = .1$



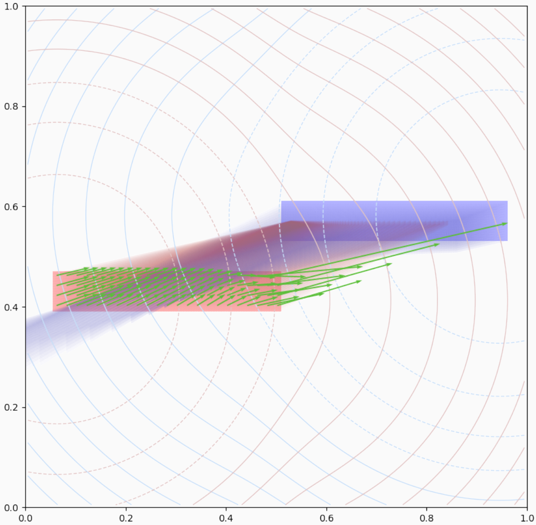
Iteration 1

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



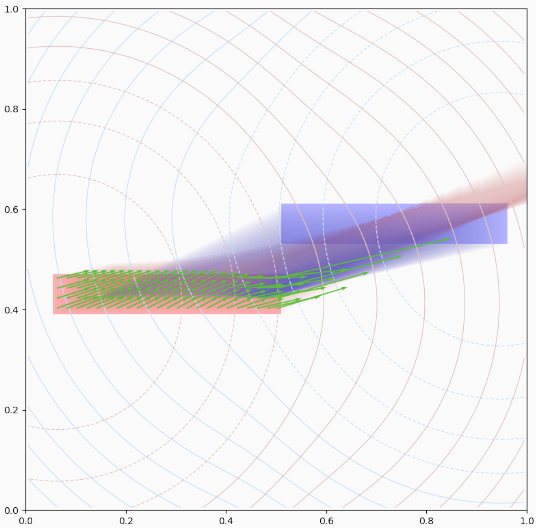
Iteration 1 ½

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



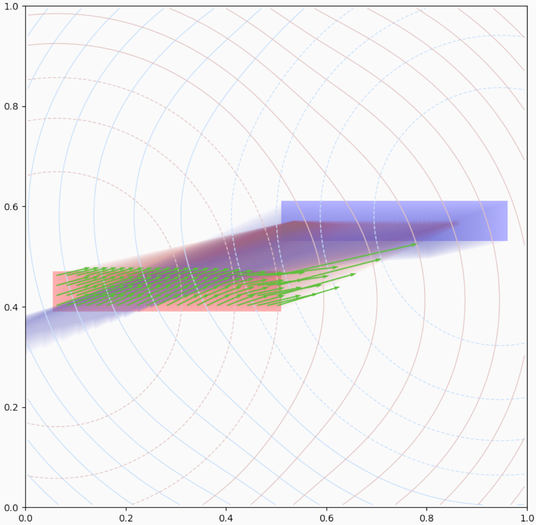
Iteration 2

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



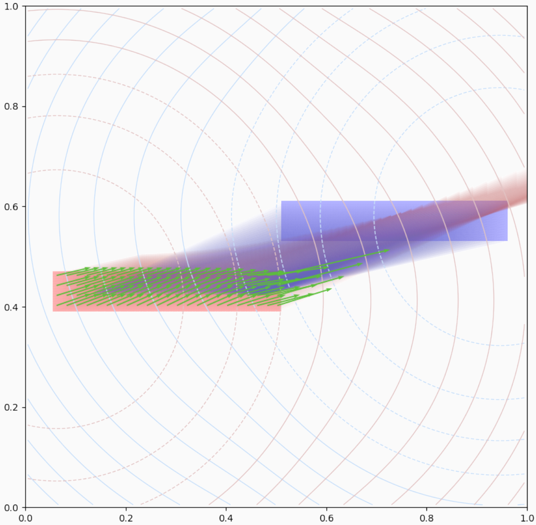
Iteration 2 ½

The Sinkhorn algorithm, in practice; with $\|x - y\|^2$ and $\sqrt{\varepsilon} = .1$



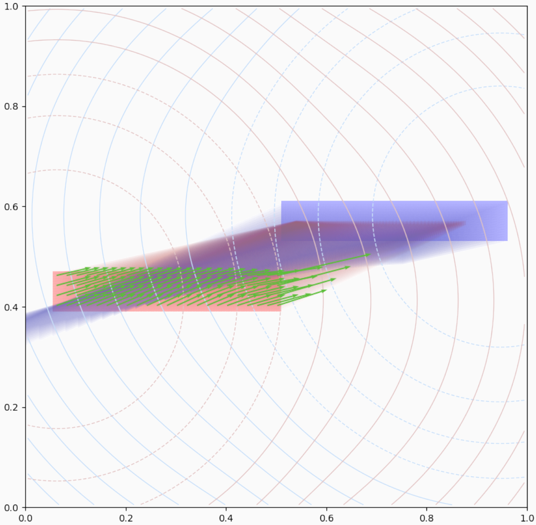
Iteration 3

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



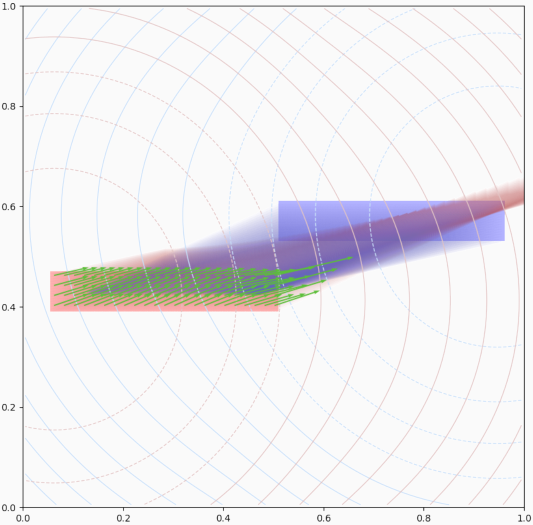
Iteration 3 ½

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



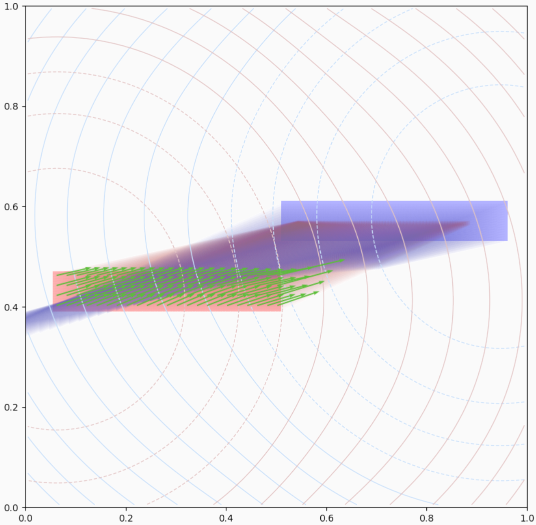
Iteration 4

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



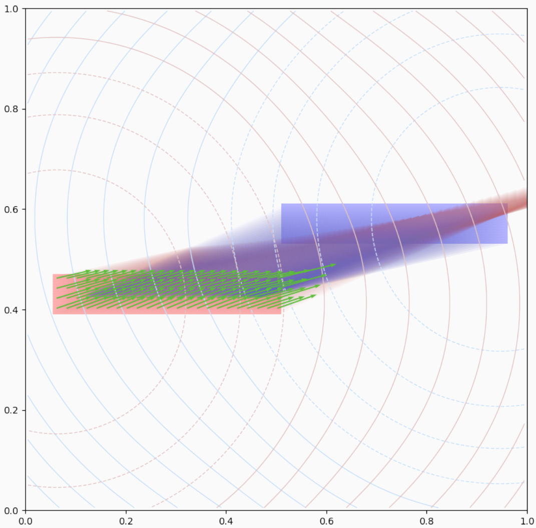
Iteration 4 ½

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



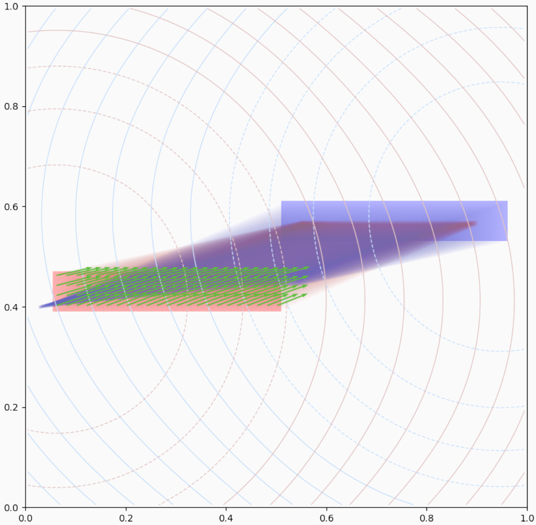
Iteration 5

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



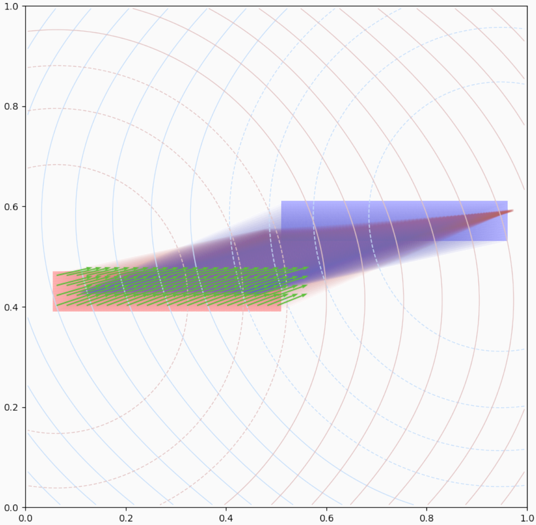
Iteration 5 ½

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



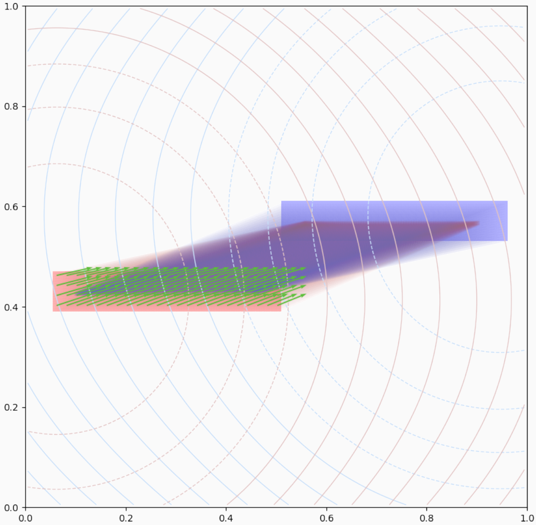
Iteration 10

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



Iteration 10 ½

The Sinkhorn algorithm, in practice; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



Iteration 20

Fact 2 : if ε is too small, the Sinkhorn algorithm does not converge

If $\varepsilon = 0$: the Sinkhorn loop gets **stuck** after two iterations.

Fact 2 : if ε is too small, the Sinkhorn algorithm does not converge

If $\varepsilon = 0$: the Sinkhorn loop gets **stuck** after two iterations.

If $\varepsilon > 0$: it is a fixed-point iteration that converges linearly...

Fact 2 : if ε is too small, the Sinkhorn algorithm does not converge

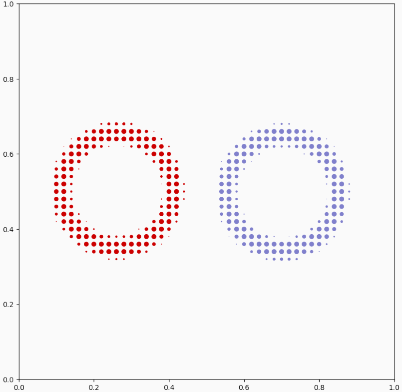
If $\varepsilon = 0$: the Sinkhorn loop gets **stuck** after two iterations.

If $\varepsilon > 0$: it is a fixed-point iteration that converges linearly...

But even in simple cases, we only converge in $O\big((1-\varepsilon)^n\big)$:
Computing a true Wasserstein distance OT_0 is out-of-reach.

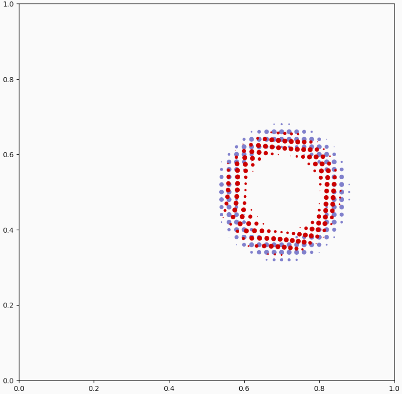
Fact 3 : if $\varepsilon > 0$, OT_ε is *not* a valid divergence

Registrating circles, $C(x,y) = \|x-y\|^2$, $\sqrt{\varepsilon} = 0.1$:



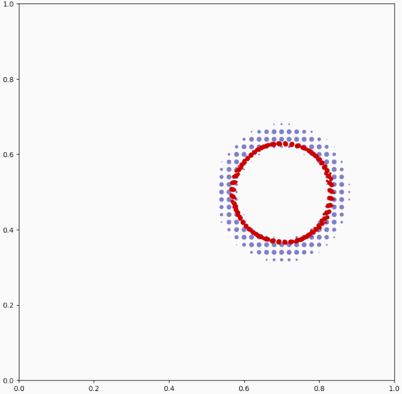
Fact 3 : if $\varepsilon > 0$, OT_ε is *not* a valid divergence

Registrating circles, $C(x, y) = \|x - y\|^2$, $\quad \sqrt{\varepsilon} = 0.1$:



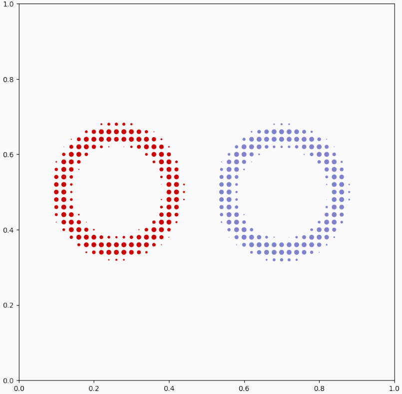
Fact 3 : if $\varepsilon > 0$, OT_ε is *not* a valid divergence

Registrating circles, $C(x,y) = \|x-y\|^2$, $\sqrt{\varepsilon} = 0.1$:



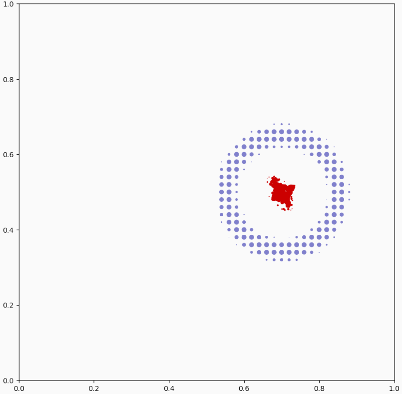
Fact 3 : if $\varepsilon > 0$, OT_ε is *not* a valid divergence

Registrating circles, $C(x, y) = \|x - y\|^2$, $\sqrt{\varepsilon} = 0.2$:



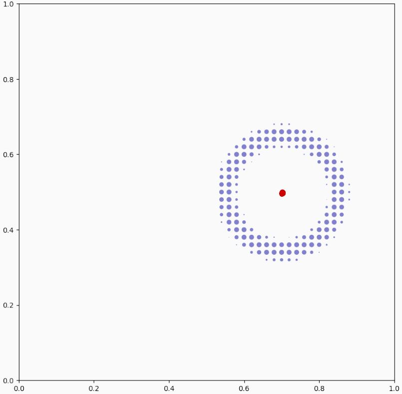
Fact 3 : if $\varepsilon > 0$, OT_ε is *not* a valid divergence

Registrating circles, $C(x,y) = \|x - y\|^2$, $\sqrt{\varepsilon} = 0.2$:



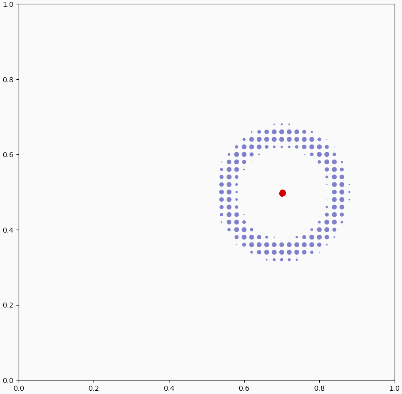
Fact 3 : if $\varepsilon > 0$, OT_ε is *not* a valid divergence

Registrating circles, $C(x, y) = \|x - y\|^2$, $\sqrt{\varepsilon} = 0.2$:



Fact 3 : if $\varepsilon > 0$, OT_ε is *not* a valid divergence

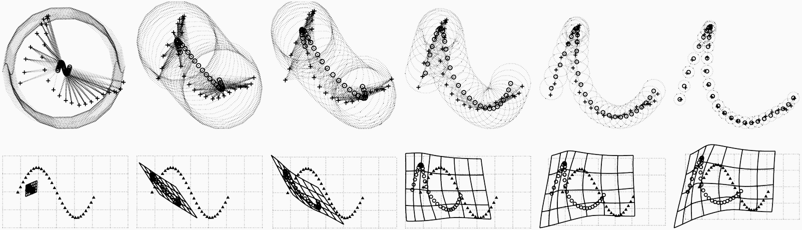
Registrating circles, $C(x, y) = \|x - y\|^2$, $\sqrt{\varepsilon} = 0.2$:



Bad news: for $0 < \varepsilon \leqslant +\infty$, we converge towards α such that

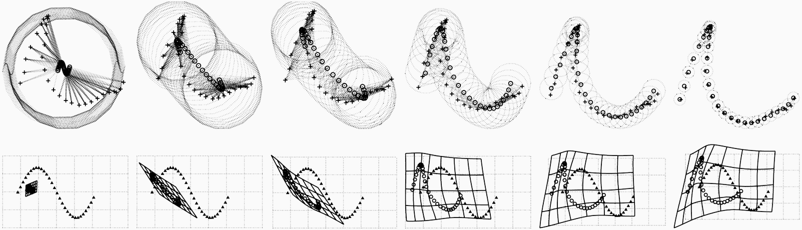
$$\mathrm{OT}_\varepsilon(\alpha, \beta) \; < \; \mathrm{OT}_\varepsilon(\beta, \beta).$$

Standard solution: use an annealing scheme



TPS-RPM algorithm, Chui and Rangarajan, CVPR 2000

Standard solution: use an annealing scheme



TPS-RPM algorithm, Chui and Rangarajan, CVPR 2000

$\Longrightarrow$ **Expensive** and cumbersome workaround, with parameters to tune.

A new idea in 2017 : un-biased Sinkhorn divergences

$$\text{OT}_\varepsilon(\alpha, \beta) \;=\; \min_{\pi} \,\langle\, \pi \,,\, \text{C} \,\rangle \;+\; \varepsilon\, \text{KL}(\,\pi\,,\, \alpha \otimes \beta\,) \quad \longrightarrow \text{Fuzzy assignment}$$

$$\text{s.t.} \qquad \pi\, \mathbf{1} \;=\; \alpha, \qquad \pi^\intercal \mathbf{1} \;=\; \beta$$

A new idea in 2017 : un-biased Sinkhorn divergences

$$\mathrm{OT}_\varepsilon(\alpha, \beta) \;=\; \min_\pi \,\langle\, \pi \,,\, \mathrm{C} \,\rangle \;+\; \varepsilon\, \mathrm{KL}(\, \pi \,,\, \alpha \otimes \beta \,) \quad \longrightarrow \text{ Fuzzy assignment}$$

$$\text{s.t.} \qquad \pi\, \mathbf{1} \;=\; \alpha, \qquad \pi^\intercal \mathbf{1} \;=\; \beta$$

$$\mathrm{OT}_\varepsilon(\alpha, \beta) \qquad \xrightarrow{\;\varepsilon \to +\infty\;} \qquad \langle\, \alpha \otimes \beta \,,\, \mathrm{C} \,\rangle \;=\; \langle\, \alpha \,,\, \mathrm{C} \star \beta \,\rangle$$

A new idea in 2017 : un-biased Sinkhorn divergences

$$\mathrm{OT}_\varepsilon(\alpha,\beta) \;=\; \min_\pi \,\langle\, \pi\,,\, \mathrm{C}\,\rangle \;+\; \varepsilon\,\mathrm{KL}(\,\pi\,,\,\alpha\otimes\beta\,) \quad\longrightarrow \text{Fuzzy assignment}$$

$$\text{s.t.} \qquad \pi\,\mathbf{1} \;=\; \alpha, \qquad \pi^\intercal\mathbf{1} \;=\; \beta$$

$$\mathrm{OT}_\varepsilon(\alpha,\beta) \quad \xrightarrow{\;\varepsilon\,\to\,+\infty\;} \quad \langle\,\alpha\otimes\beta\,,\,\mathrm{C}\,\rangle \;=\; \langle\,\alpha\,,\,\mathrm{C}\star\beta\,\rangle$$

Define the **Sinkhorn divergence** [Ramdas et al., 2017]:

$$\mathrm{S}_\varepsilon(\alpha,\beta) \;=\; \mathrm{OT}_\varepsilon(\alpha,\beta) \,-\, \tfrac{1}{2}\mathrm{OT}_\varepsilon(\alpha,\alpha) \,-\, \tfrac{1}{2}\mathrm{OT}_\varepsilon(\beta,\beta)$$

A new idea in 2017 : un-biased Sinkhorn divergences

$$\mathrm{OT}_\varepsilon(\alpha,\beta) \;=\; \min_{\pi} \langle\, \pi \,,\, \mathrm{C} \,\rangle \;+\; \varepsilon\, \mathrm{KL}(\,\pi\,,\,\alpha\otimes\beta\,) \quad \longrightarrow \text{Fuzzy assignment}$$

$$\text{s.t.} \qquad \pi\,\mathbf{1} \;=\; \alpha, \qquad \pi^\intercal\mathbf{1} \;=\; \beta$$

$$\mathrm{OT}_\varepsilon(\alpha,\beta) \qquad \xrightarrow{\varepsilon\,\to\,+\infty} \qquad \langle\, \alpha\otimes\beta\,,\,\mathrm{C}\,\rangle \;=\; \langle\, \alpha\,,\,\mathrm{C}\star\beta\,\rangle$$

Define the **Sinkhorn divergence** [Ramdas et al., 2017]:

$$\mathrm{S}_\varepsilon(\alpha,\beta) \;=\; \mathrm{OT}_\varepsilon(\alpha,\beta) - \tfrac{1}{2}\mathrm{OT}_\varepsilon(\alpha,\alpha) - \tfrac{1}{2}\mathrm{OT}_\varepsilon(\beta,\beta)$$

$$\mathrm{Wasserstein}_{+\mathrm{C}}(\alpha,\beta) \xleftarrow{\varepsilon\to 0} \mathrm{S}_\varepsilon(\alpha,\beta) \xrightarrow{\varepsilon\to+\infty} \mathrm{Kernel}_{-\mathrm{C}}(\alpha,\beta)$$

A new idea in 2017 : un-biased Sinkhorn divergences

$$\mathrm{OT}_\varepsilon(\alpha, \beta) \;=\; \min_{\pi} \langle\, \pi \,,\, \mathrm{C} \,\rangle \;+\; \varepsilon\, \mathrm{KL}(\, \pi \,,\, \alpha \otimes \beta \,) \quad \longrightarrow \text{ Fuzzy assignment}$$

$$\text{s.t.} \qquad \pi\, \mathbf{1} \;=\; \alpha, \qquad \pi^\top \mathbf{1} \;=\; \beta$$

$$\mathrm{OT}_\varepsilon(\alpha, \beta) \quad \xrightarrow{\;\varepsilon \,\to\, +\infty\;} \quad \langle\, \alpha \otimes \beta \,,\, \mathrm{C} \,\rangle \;=\; \langle\, \alpha \,,\, \mathrm{C} \star \beta \,\rangle$$

Define the **Sinkhorn divergence** [Ramdas et al., 2017]:

$$\mathrm{S}_\varepsilon(\alpha, \beta) \;=\; \mathrm{OT}_\varepsilon(\alpha, \beta) - \tfrac{1}{2}\mathrm{OT}_\varepsilon(\alpha, \alpha) - \tfrac{1}{2}\mathrm{OT}_\varepsilon(\beta, \beta)$$

$$\mathrm{Wasserstein}_{+\mathrm{C}}(\alpha, \beta) \;\xleftarrow{\;\varepsilon\to 0\;}\; \mathrm{S}_\varepsilon(\alpha, \beta) \;\xrightarrow{\;\varepsilon\to+\infty\;}\; \mathrm{Kernel}_{-\mathrm{C}}(\alpha, \beta)$$

In practice, S_ε is “good enough” for ML applications
[Genevay et al., 2018, Salimans et al., 2018, Sanjabi et al., 2018].

In our paper: theoretical guarantees

Theorem (F., Séjourné, Vialard, Amari, Trouvé, Peyré; 2018)

For all probability measures α, β *and regularization* $\varepsilon > 0$ *:*

In our paper: theoretical guarantees

Theorem (F., Séjourné, Vialard, Amari, Trouvé, Peyré; 2018)

For all probability measures α, β *and regularization* $\varepsilon > 0$ *:*

$$0 \leqslant S_\varepsilon(\alpha, \beta) \quad \textit{with equality iff.} \quad \alpha = \beta$$

In our paper: theoretical guarantees

Theorem (F., Séjourné, Vialard, Amari, Trouvé, Peyré; 2018)

For all probability measures α, β *and regularization* $\varepsilon > 0$ *:*

$$0 \leqslant S_\varepsilon(\alpha, \beta) \quad \text{with equality iff.} \quad \alpha = \beta$$

$$\alpha \mapsto S_\varepsilon(\alpha, \beta) \quad \text{is convex and differentiable}$$

In our paper: theoretical guarantees

Theorem (F., Séjourné, Vialard, Amari, Trouvé, Peyré; 2018)

For all probability measures α, β *and regularization* $\varepsilon > 0$ *:*

$$0 \leqslant S_\varepsilon(\alpha, \beta) \quad \text{with equality iff.} \quad \alpha = \beta$$

$$\alpha \mapsto S_\varepsilon(\alpha, \beta) \quad \text{is convex and differentiable}$$

*These results can be generalized to arbitrary **feature** spaces*
– e.g. (position,orientation,curvature).

In our paper: theoretical guarantees

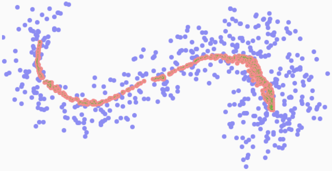
Theorem (F., Séjourné, Vialard, Amari, Trouvé, Peyré; 2018)

For all probability measures α, β *and regularization* $\varepsilon > 0$:

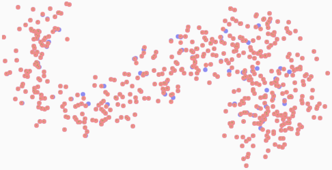
$$0 \leqslant \mathrm{S}_\varepsilon(\alpha, \beta) \quad \textit{with equality iff.} \quad \alpha = \beta$$

$$\alpha \mapsto \mathrm{S}_\varepsilon(\alpha, \beta) \quad \textit{is convex and differentiable}$$

These results can be generalized to arbitrary ***feature*** *spaces – e.g. (position,orientation,curvature).*



Loss $= \mathrm{OT}_\varepsilon$



Loss $= \mathrm{S}_\varepsilon$

Conclusion

A significant result

The **true** OT_0 problem is hard.

A significant result

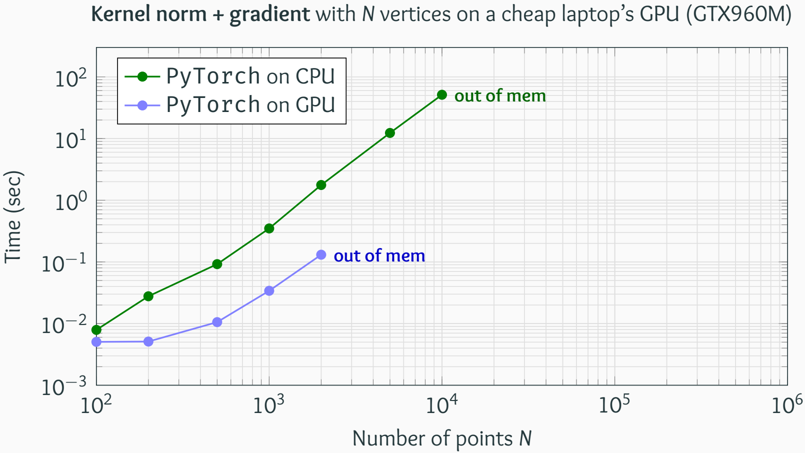
The **true** OT_0 problem is hard.
Approximating it with **subsampling** or **smoothing** is easy:
this is what **SoftAssign** is all about.

A significant result

The **true** OT_0 problem is hard.
Approximating it with **subsampling** or **smoothing** is easy:
this is what **SoftAssign** is all about.

Remarkably, $\mathrm{S}_\varepsilon(\alpha, \beta)$ is a cheap approximation of $\mathrm{OT}_0(\alpha, \beta)$
that defines a **positive definite** cost between the **full samples**.
It is the first known way of doing so.

KErnel OPerationS, with autodiff, without memory overflows

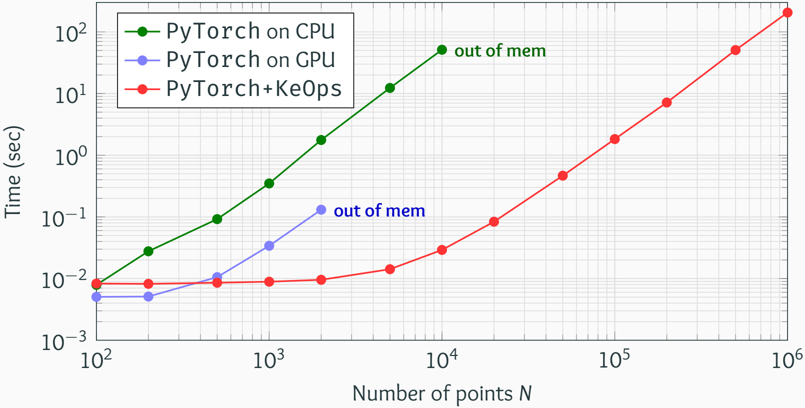


KErnel OPerationS, with autodiff, without memory overflows

$\Longrightarrow$ `pip install pykeops` $\Longleftarrow$

(Thanks Benjamin and Joan!)

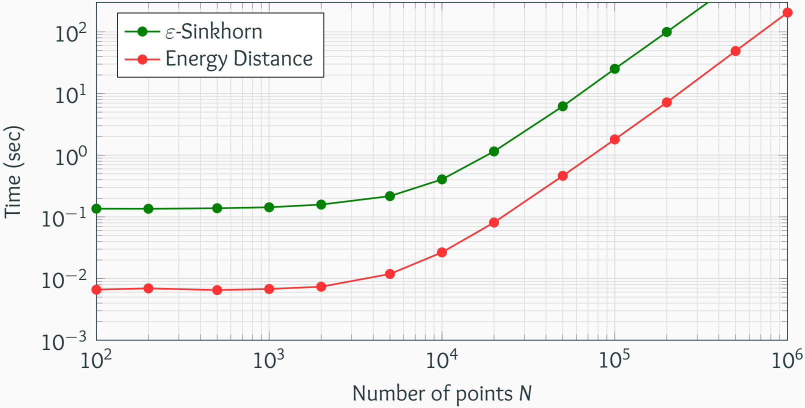
Kernel norm + gradient with *N* vertices on a cheap laptop's GPU (GTX960M)



$\Longrightarrow$ `pip install pykeops` $\Longleftarrow$

(Thanks Benjamin and Joan!)

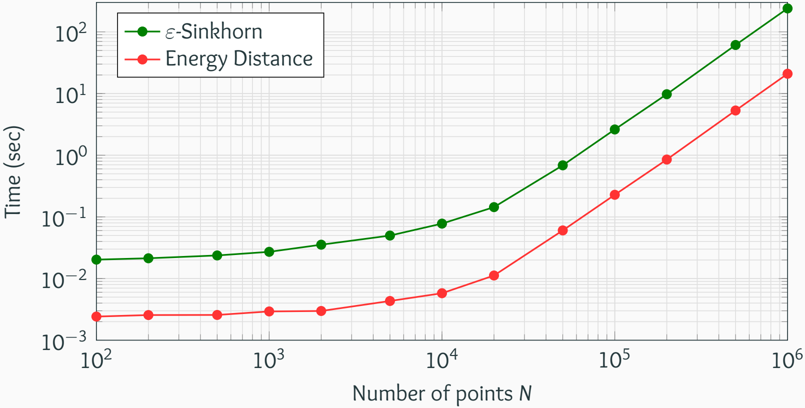
Fidelity + gradient with N vertices on a **cheap laptop**'s GPU (GTX960M)



$\Longrightarrow$ `pip install pykeops` $\Longleftarrow$

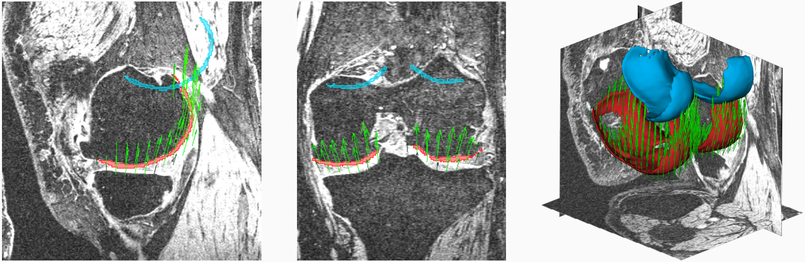
(Thanks Benjamin and Joan!)

Fidelity + gradient with N vertices on a **high-end** GPU (Tesla P100)



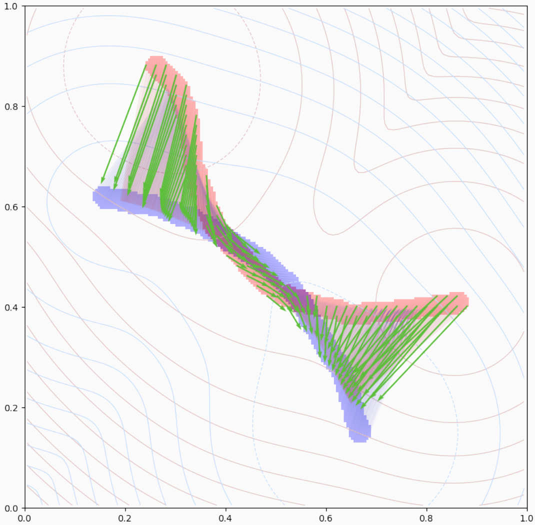
We provide a reference PyTorch implementation

`github.com/jeanfeydy/global-divergences`.



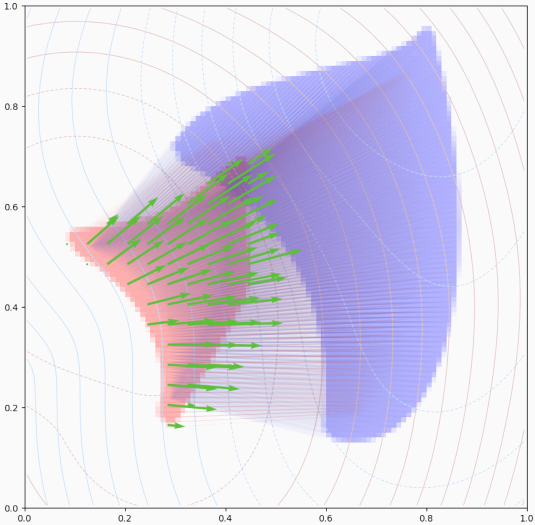
Gradient of the Energy Distance, computed in 0.5s on my laptop.
Data from the OsteoArthritris Initiative:
52,319 and 34,966 voxels out of a 192-192-160 volume.

The ε-Sinkhorn divergence; with $\|x - y\|^2$ and $\sqrt{\varepsilon} = .1$



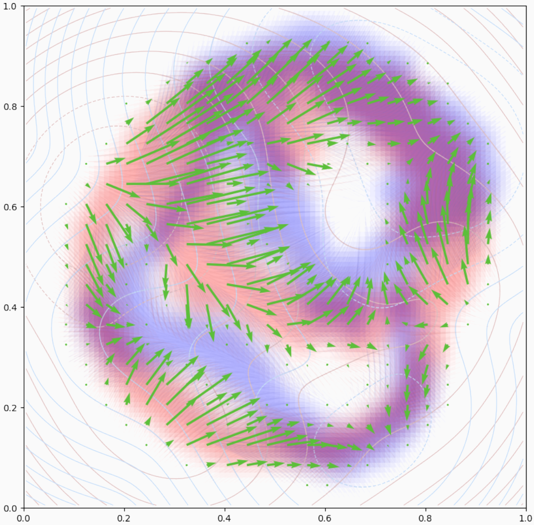
A high-quality gradient.

The ε-Sinkhorn divergence; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



A high-quality gradient.

The ε-Sinkhorn divergence; with $\|x-y\|^2$ and $\sqrt{\varepsilon} = .1$



A high-quality gradient?

(Data from the Spectral Log-Demons paper.)

Conclusion

Global, **geometry-aware** loss functions are easy to compute.

Conclusion

Global, **geometry-aware** loss functions are easy to compute.

- Try using $k(x, y) = -\|x - y\|$!

Conclusion

Global, **geometry-aware** loss functions are easy to compute.

- Try using $k(x, y) = -\|x - y\|$!
- Remove the **entropic bias** from the SoftAssign algorithm!

Conclusion

Global, **geometry-aware** loss functions are easy to compute.

- Try using $k(x, y) \; = \; -\|x - y\|$!
- Remove the **entropic bias** from the SoftAssign algorithm!
- Sinkhorn $=$ Hausdorff + mass **spreading** constraint
  $\simeq$ best you can do without topology or landmarks
  $\simeq$ 20-50 convolutions through the data
  $\rightarrow$ Is it worth it?

Conclusion

Our work:

- Miccai2017 : proof of concept

Conclusion

Our work:

- Miccai2017 : proof of concept
- ShapeMi2018/AiStats2019 :
  - link with statistics and **computer graphics**
  - reference **implementation** on sparse data
  - theoretical **guarantees**

Conclusion

Our work:

- Miccai2017 : proof of concept
- ShapeMi2018/AiStats2019 :
  - link with statistics and **computer graphics**
  - reference **implementation** on sparse data
  - theoretical **guarantees**
- 2019 :
  - unbalanced formulation, gestion of **outliers**
  - **evaluation** in varied settings
  - separable **volumetric** implementation

Conclusion

Open questions:

- **Combinatorial** interpretation of the Sinkhorn iterations?

Conclusion

Open questions:

- **Combinatorial** interpretation of the Sinkhorn iterations?
- Link between S_ε and Sobolev **distances**?

Conclusion

Open questions:

- **Combinatorial** interpretation of the Sinkhorn iterations?
- Link between S_ε and Sobolev **distances**?
- What about **Octrees**?

Conclusion

Open questions:

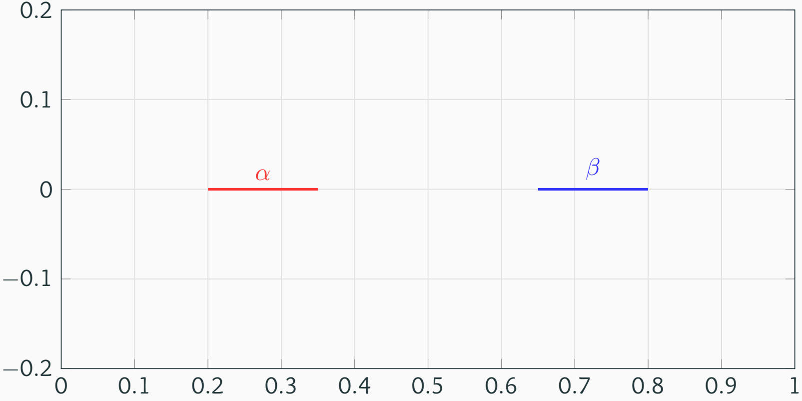
- **Combinatorial** interpretation of the Sinkhorn iterations?
- Link between S_ε and Sobolev **distances**?
- What about **Octrees**?
- Interest in the **CVPR/SIGGRAPH** communities?

Thank you for your attention.

Any questions ?

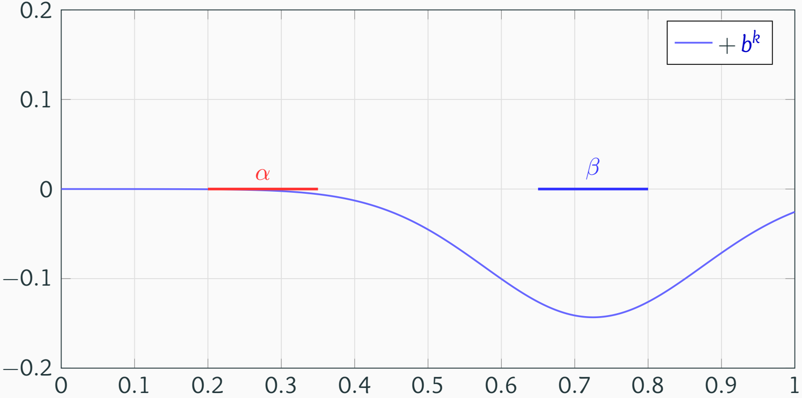
The registration flows along the gradient of $b^k - a^k$

$$k(x - y) \;=\; \exp(-\|x - y\|^2 / .2^2)$$



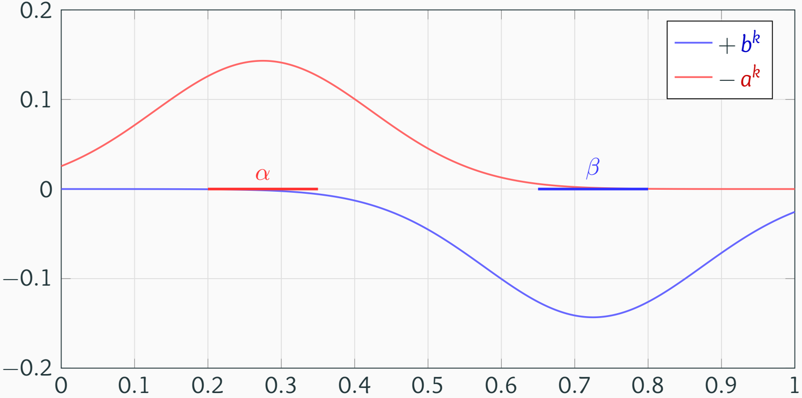
The registration flows along the gradient of $b^k - a^k$

$$k(x-y) \;=\; \exp(-\|x-y\|^2/.2^2)$$

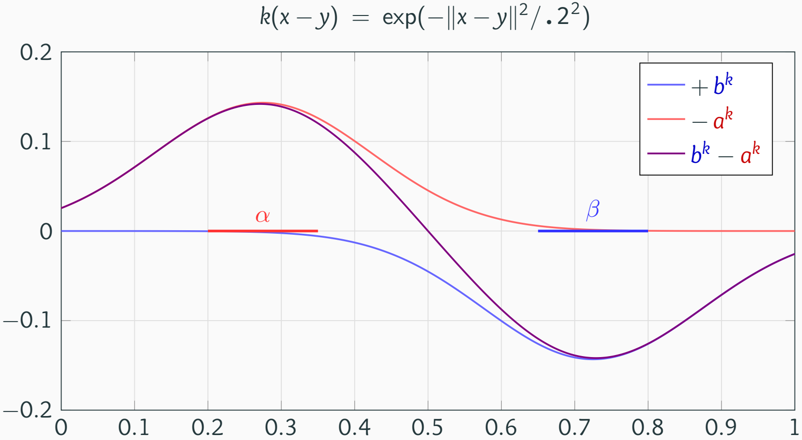


The registration flows along the gradient of $b^k - a^k$

$$k(x-y) \;=\; \exp(-\|x-y\|^2/\,.2^2)$$

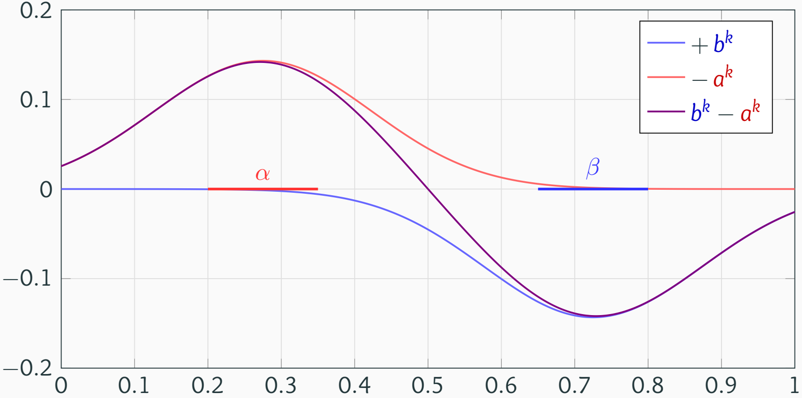


The registration flows along the gradient of $b^k - a^k$



The registration flows along the gradient of $b^k - a^k$

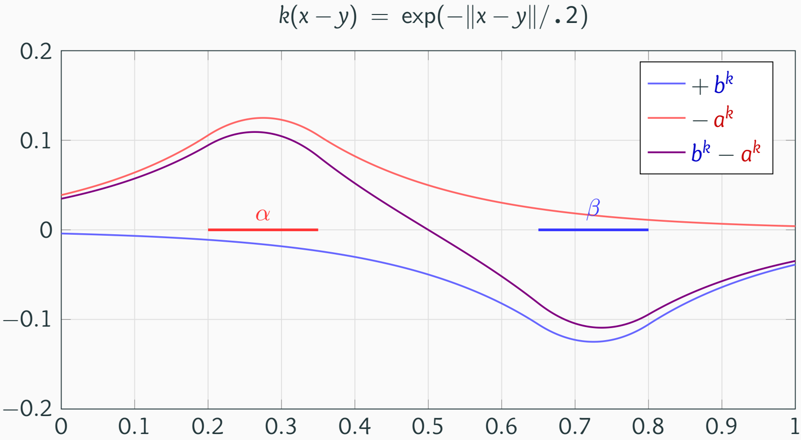
$$k(x-y) \;=\; \exp(-\|x-y\|^2/.2^2)$$



$$\mathrm{d}_k(\alpha,\beta) \;=\; \tfrac{1}{2}\big\langle\, \alpha-\beta \,|\, k\star(\alpha-\beta)\,\big\rangle$$

$$\nabla_{x_i}\mathrm{d}_k(\alpha,\beta) \;=\; \nabla\big[k\star(\alpha-\beta)\big](x_i) \;=\; \nabla b^k(x_i) - \nabla a^k(x_i)$$

The registration flows along the gradient of $b^k - a^k$

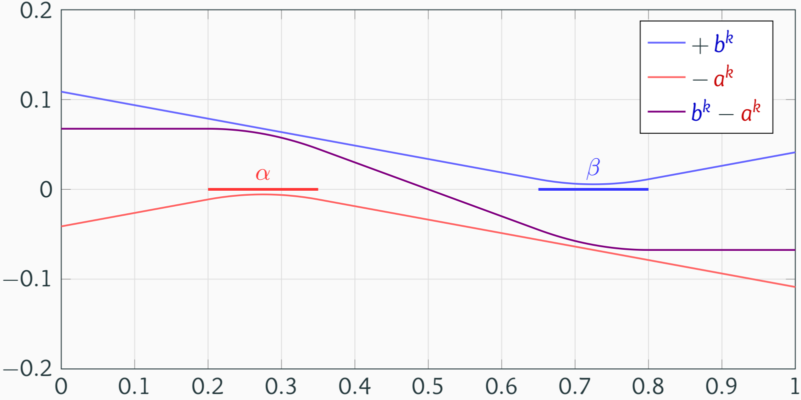


$$\mathrm{d}_k(\alpha,\beta) \;=\; \tfrac{1}{2}\langle\, \alpha-\beta \mid k\star(\alpha-\beta)\,\rangle$$

$$\nabla_{x_i}\mathrm{d}_k(\alpha,\beta) \;=\; \nabla\big[k\star(\alpha-\beta)\big](x_i) \;=\; \nabla b^k(x_i) - \nabla a^k(x_i)$$

The registration flows along the gradient of $b^k - a^k$

$$k(x-y) \;=\; -\|x-y\|$$

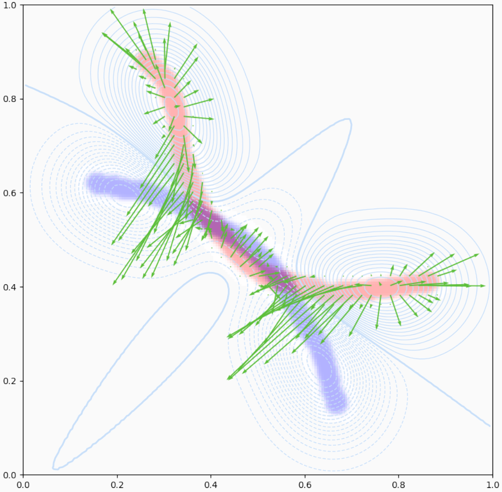


$$\mathrm{d}_k(\alpha,\beta) \;=\; \tfrac{1}{2}\big\langle\, \alpha-\beta \,\big|\, k\star(\alpha-\beta) \,\big\rangle$$

$$\nabla_{x_i}\mathrm{d}_k(\alpha,\beta) \;=\; \nabla\big[k\star(\alpha-\beta)\big](x_i) \;=\; \nabla b^k(x_i) - \nabla a^k(x_i)$$

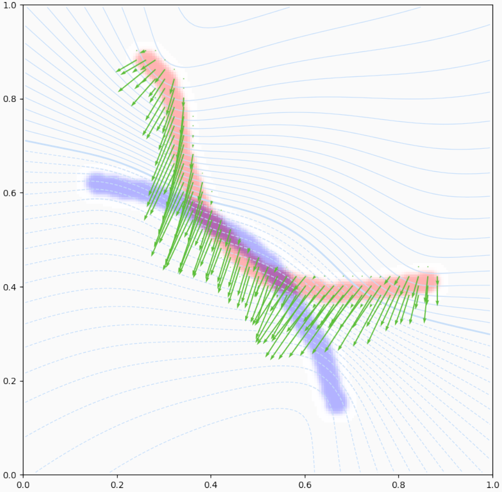
The Energy Distance is scale-invariant, robust

$$k(x-y) \;=\; \exp(-\|x-y\|^2/\,.1^2)$$



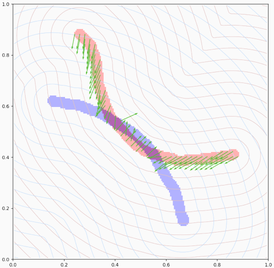
The Energy Distance is scale-invariant, robust

$$k(x - y) \;=\; -\|x - y\|$$

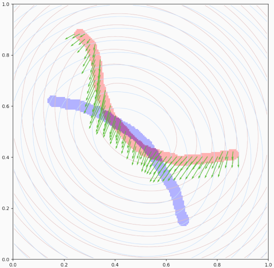


An idea from computer graphics: Hausdorff distances

The SoftMin fidelity interpolates between Hausdorff and ED

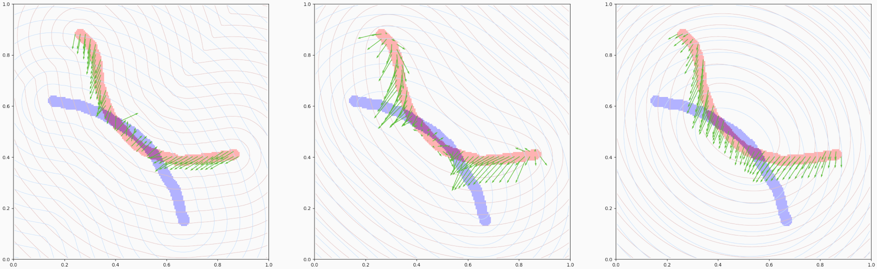


Hausdorff, min



Kernel, $\sum$

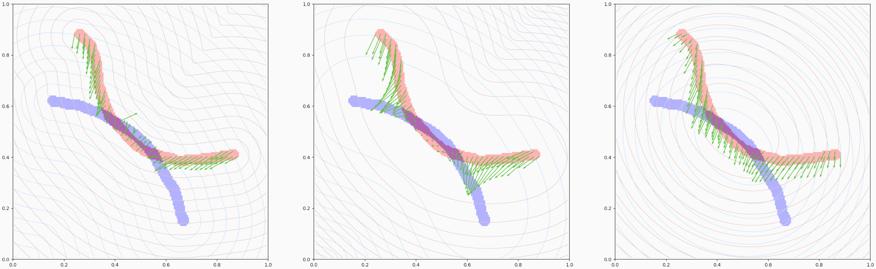
The SoftMin fidelity interpolates between Hausdorff and ED



Hausdorff, min	SoftMin, $\min^\varepsilon$	Kernel, $\sum$
$\varepsilon = 0$	$\longleftrightarrow$	$\varepsilon = +\infty$

$$\max^\varepsilon(c, d) \;\; = \;\; \varepsilon \, \log \Big(\, \exp(\tfrac{c}{\varepsilon}) + \exp(\tfrac{d}{\varepsilon}) \, \Big)$$

The SoftMin fidelity interpolates between Hausdorff and ED



Hausdorff, min	SoftMin, $\min^\varepsilon$	Kernel, $\sum$
$\varepsilon = 0$	$\longleftrightarrow$	$\varepsilon = +\infty$

$$\max^\varepsilon(c, d) \; = \; \varepsilon \log \big(\exp(\tfrac{c}{\varepsilon}) + \exp(\tfrac{d}{\varepsilon}) \big)$$

You can also use it with e.g. $\|x - y\|^2$ instead of $\|x - y\|$.

References

Our papers:

- *Global divergences between measures: from Hausdorff distance to Optimal Transport*, F., Trouvé, 2018

References

Our papers:

- *Global divergences between measures: from Hausdorff distance to Optimal Transport*, F., Trouvé, 2018
- *Sinkhorn entropies and divergences*,
  F., Séjourné, Vialard, Amari, Trouvé, Peyré, 2018

References

Our papers:

- *Global divergences between measures: from Hausdorff distance to Optimal Transport*, F., Trouvé, 2018
- *Sinkhorn entropies and divergences*,
  F., Séjourné, Vialard, Amari, Trouvé, Peyré, 2018
- *Optimal Transport for diffeomorphic registration*,
  F., Charlier, Vialard, Peyré, 2017

References i

Chizat, L., Peyré, G., Schmitzer, B., and Vialard, F.-X. (2018).
Unbalanced optimal transport: Dynamic and kantorovich formulations.
Journal of Functional Analysis, 274(11):3090–3123.

Cuturi, M. (2013).
Sinkhorn distances: Lightspeed computation of optimal transport.
In *Advances in neural information processing systems*, pages 2292–2300.

References ii

Genevay, A., Peyre, G., and Cuturi, M. (2018).
Learning generative models with sinkhorn divergences.
In Storkey, A. and Perez-Cruz, F., editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1608–1617. PMLR.

Kaltenmark, I., Charlier, B., and Charon, N. (2017).
A general framework for curve and surface comparison and registration with oriented varifolds.
In *Computer Vision and Pattern Recognition (CVPR)*.

Peyré, G. and Cuturi, M. (2018).
Computational optimal transport.
arXiv preprint arXiv:1803.00567.

Ramdas, A., Trillos, N. G., and Cuturi, M. (2017).
On wasserstein two-sample testing and related families of nonparametric tests.
Entropy, 19(2).

Salimans, T., Zhang, H., Radford, A., and Metaxas, D. (2018).
Improving GANs using optimal transport.
arXiv preprint arXiv:1803.05573.

Sanjabi, M., Ba, J., Razaviyayn, M., and Lee, J. D. (2018).
On the convergence and robustness of training GANs with regularized optimal transport.
arXiv preprint arXiv:1802.08249.