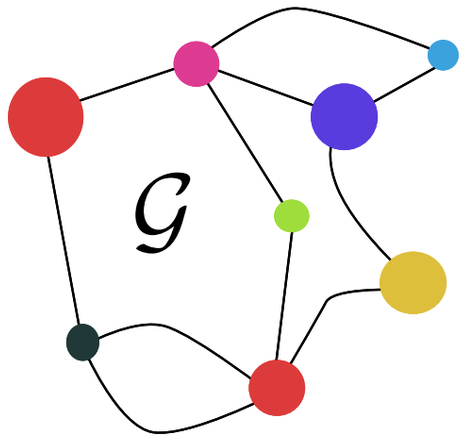
Learning on incomparable spaces using Optimal Transport



Titouan Vayer

Post-Doc ENS LYON

TAU

25/11/2021

Remi Gribonval

Nicolas Courty

Rémi Flamary

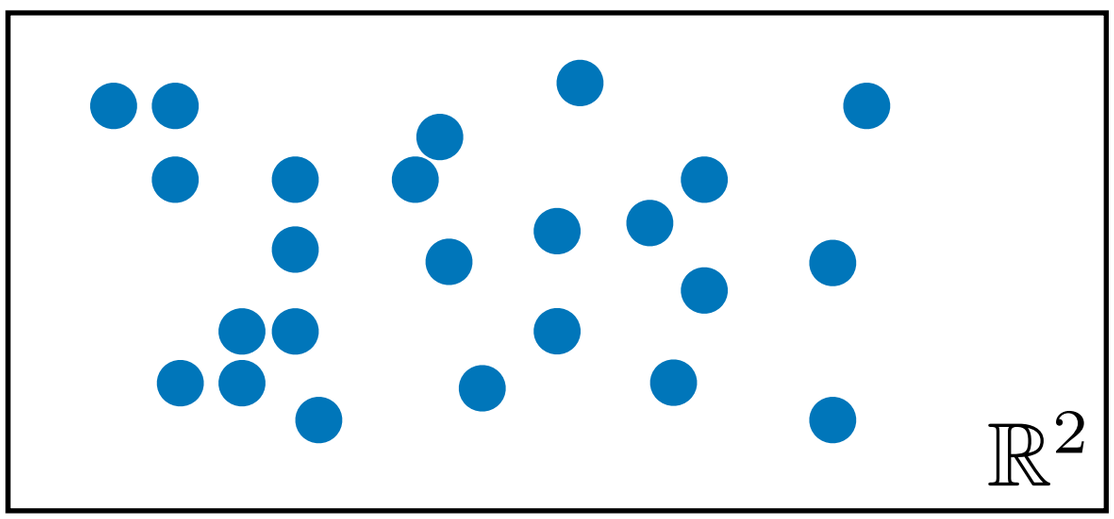
Cédric Vincent-Cuaz

Ievgen Redko

Marco Cornelli

In short:

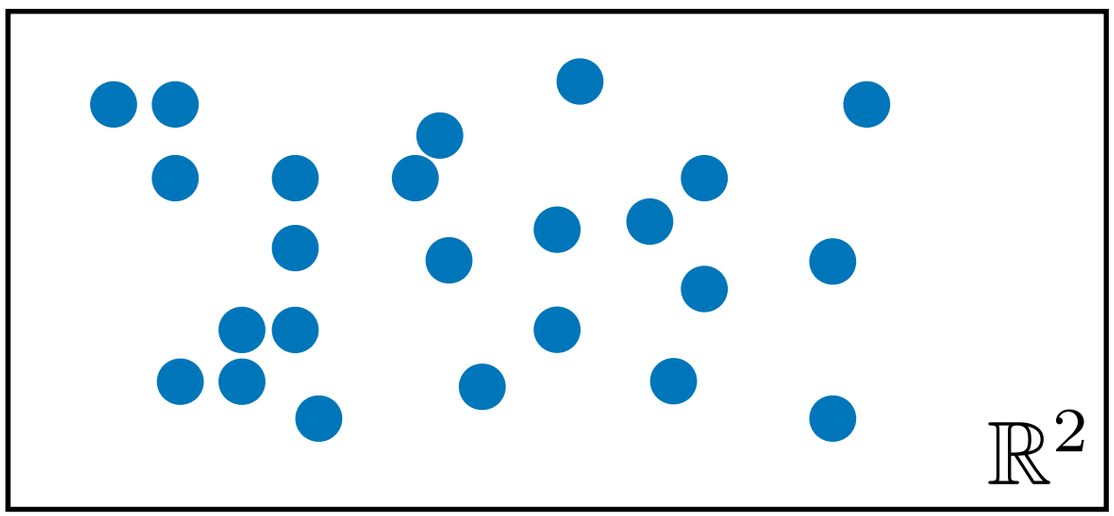
Machine Learning: Learn to make decision from **data**



In short:

Machine Learning: Learn to make decision from **data**

- **How to represent data?** → **Mathematical representation**
- **How to operate on them?** → **Tools which build upon this representation**



In short:

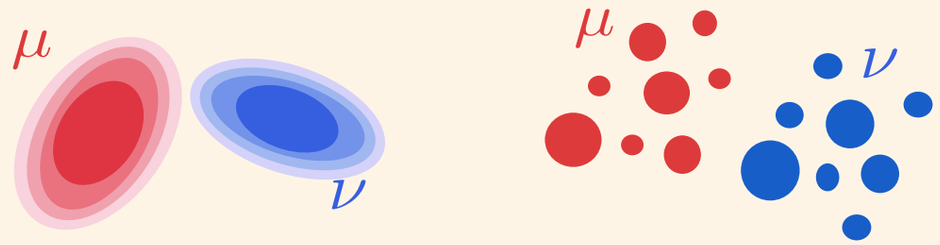
Machine Learning: Learn to make decision from **data**

How to represent data?

How to operate on them?

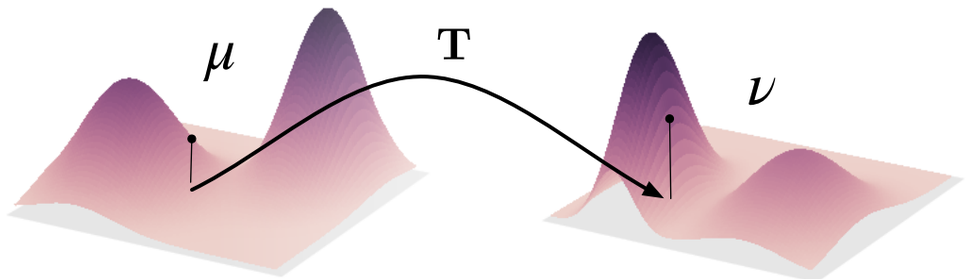
Mathematical representation

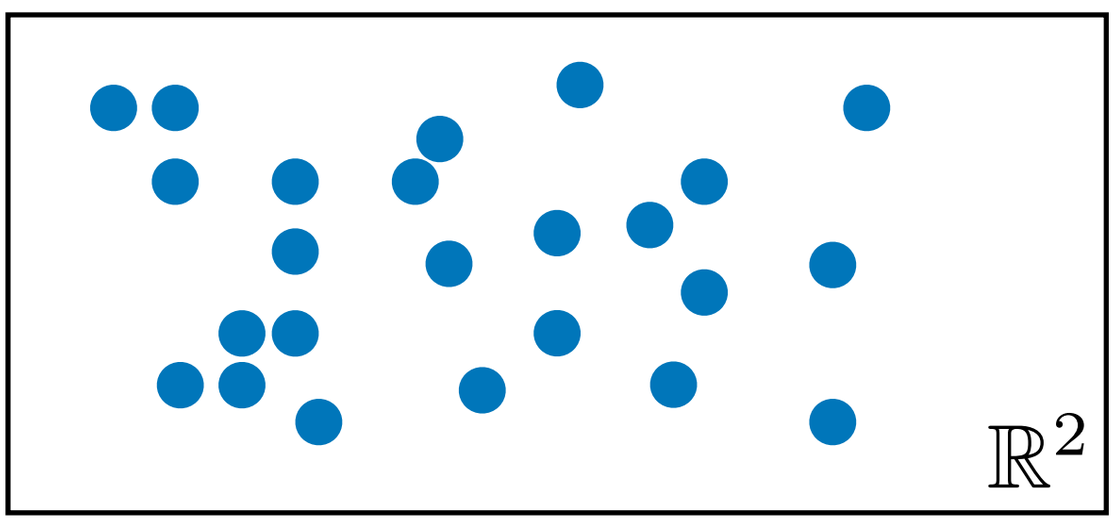
As probability distributions

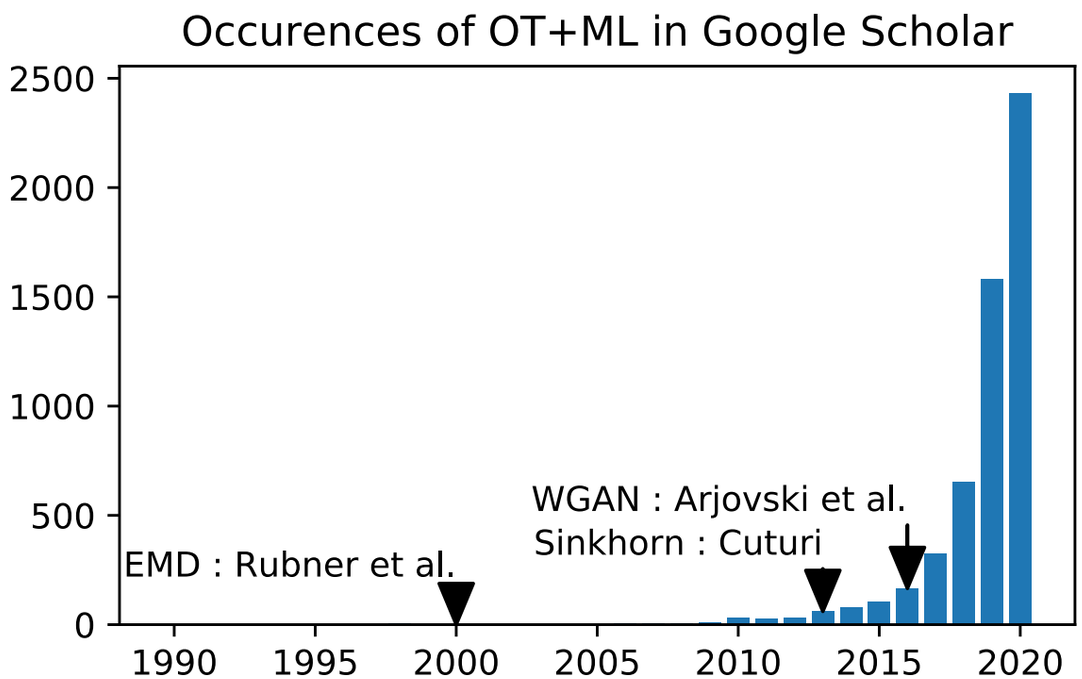


Tools which build upon this representation

Optimal Transport theory







In short:

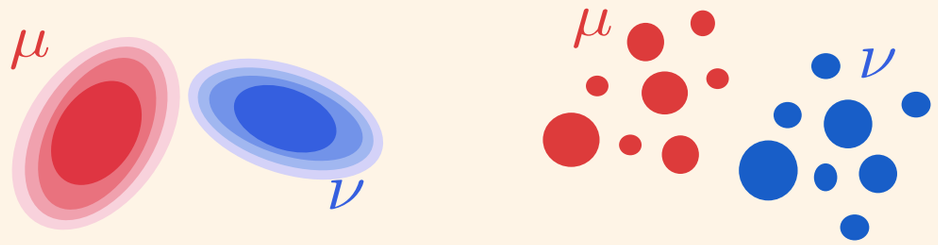
Machine Learning: Learn to make decision from **data**

How to represent data?

How to operate on them?

Mathematical representation

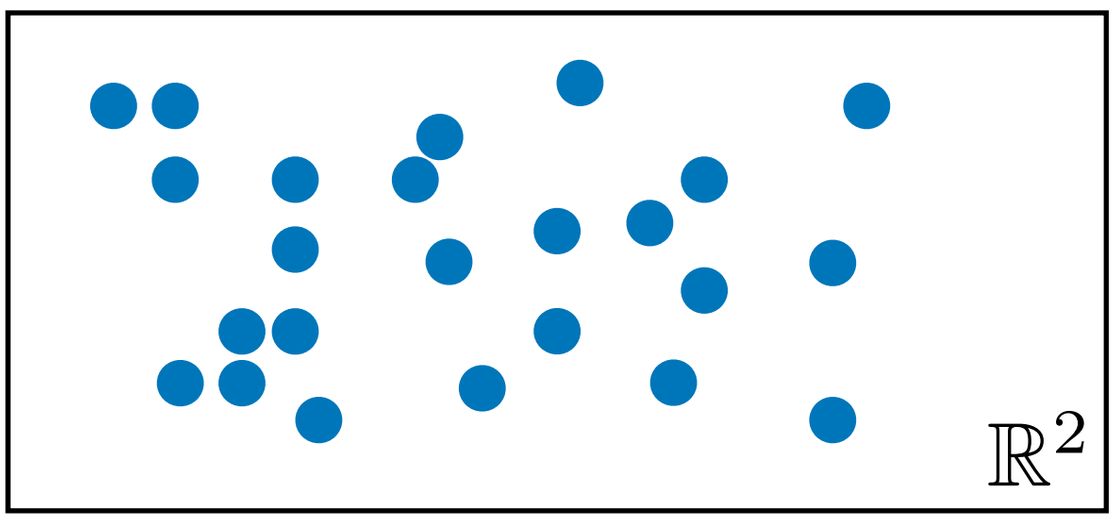
As probability distributions



Tools which build upon this representation

Optimal Transport theory

Particularly challenging: highly structured data, heterogeneous spaces



In short:

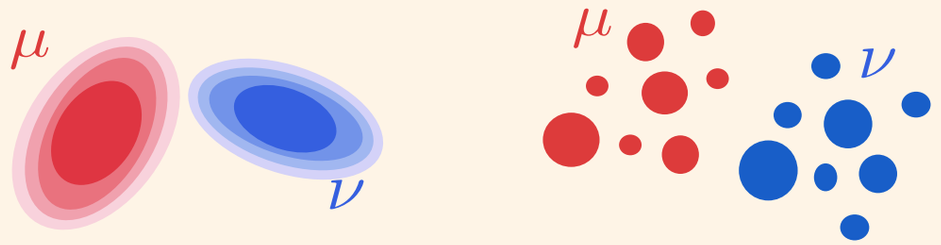
Machine Learning: Learn to make decision from **data**

How to represent data?

How to operate on them?

Mathematical representation

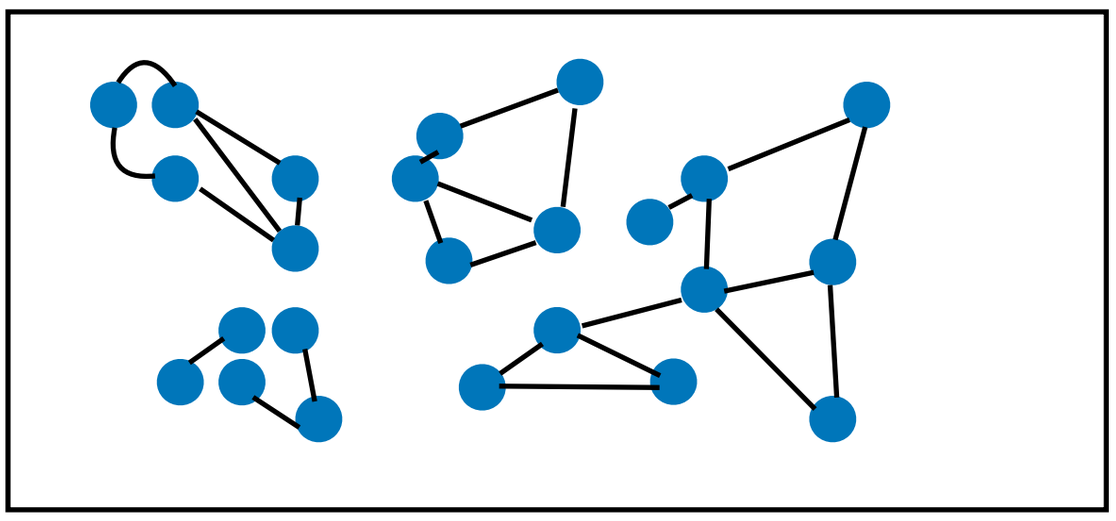
As probability distributions

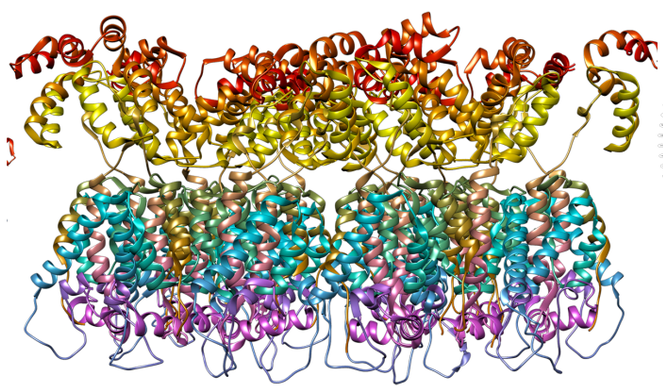


Tools which build upon this representation

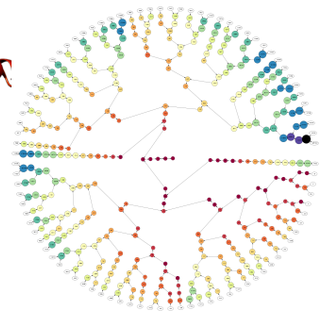
Optimal Transport theory

Particularly challenging: highly structured data, heterogeneous spaces





molecules, sequences..



graphs

In short:

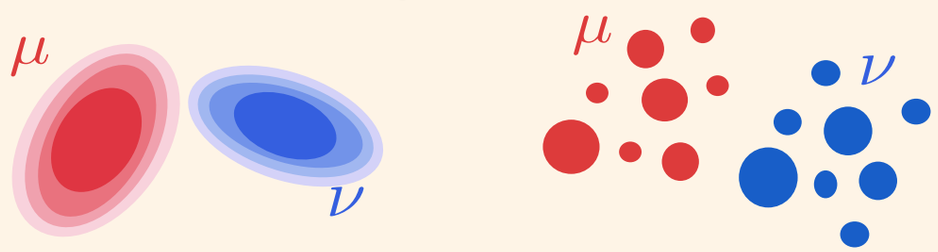
Machine Learning: Learn to make decision from **data**

How to represent data?

How to operate on them?

Mathematical representation

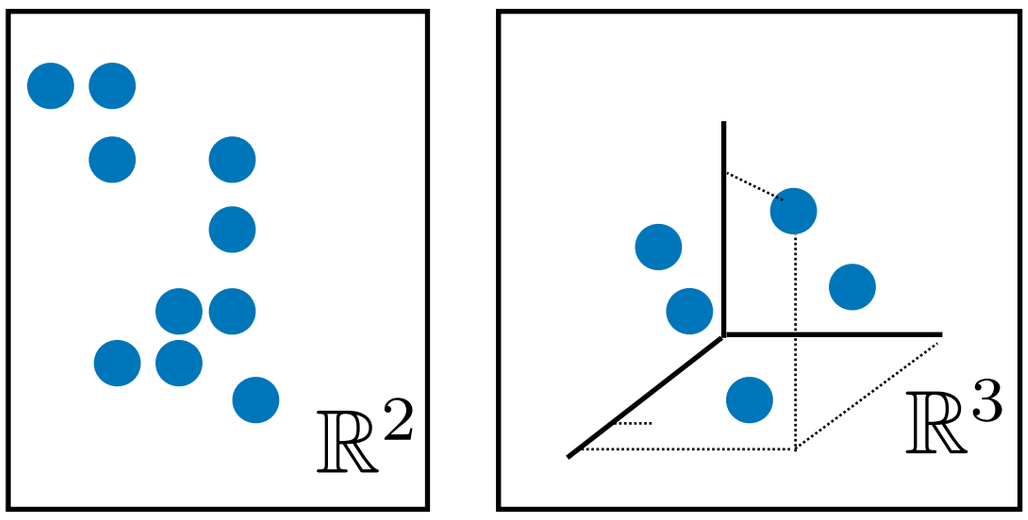
As probability distributions

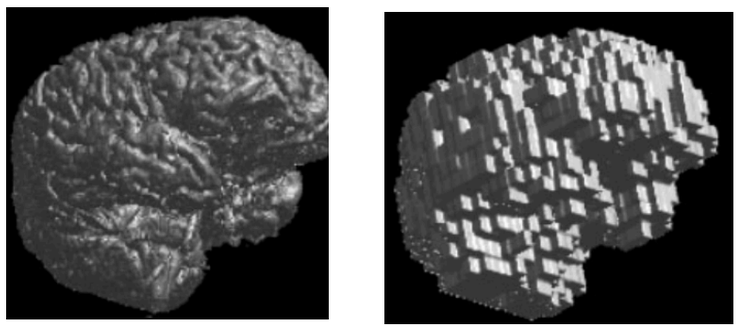


Tools which build upon this representation

Optimal Transport theory

Particularly challenging: highly structured data, **heterogeneous spaces**





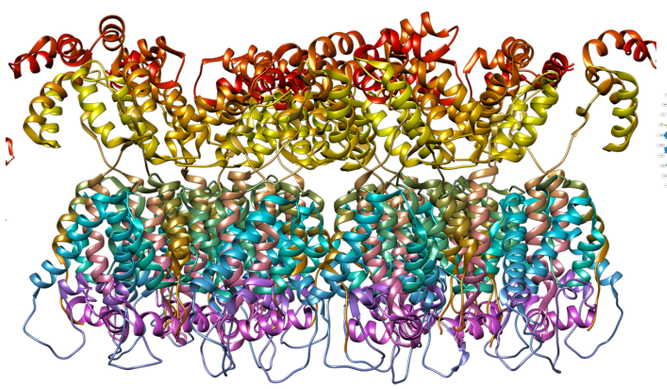
high & low resolution images

In short:

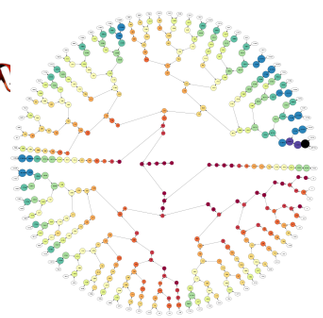
Machine Learning: Learn to make decision from **data**

- **How to represent data?**
- **How to operate on them?**

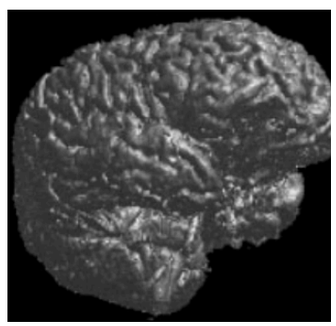
Particularly challenging: highly structured data, heterogeneous spaces

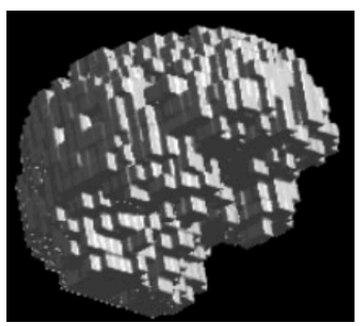


molecules, sequences..



graphs

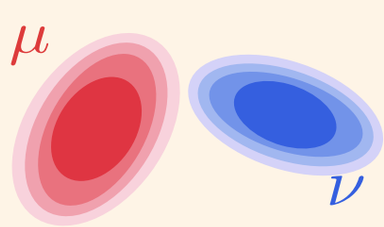


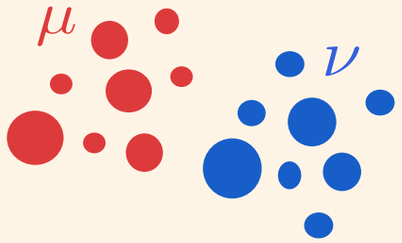


high & low resolution images

Mathematical representation

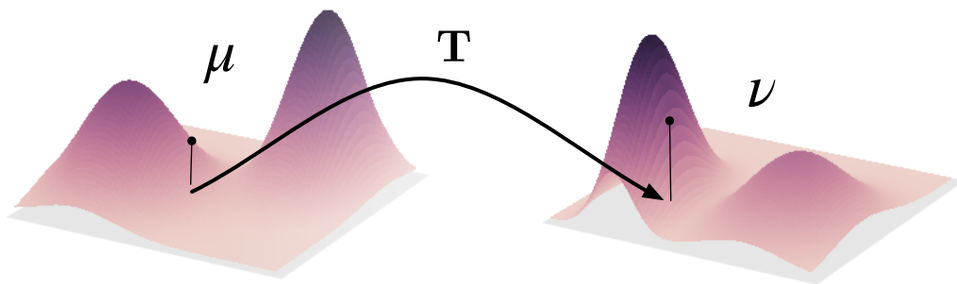
As probability distributions





Tools which build upon this representation

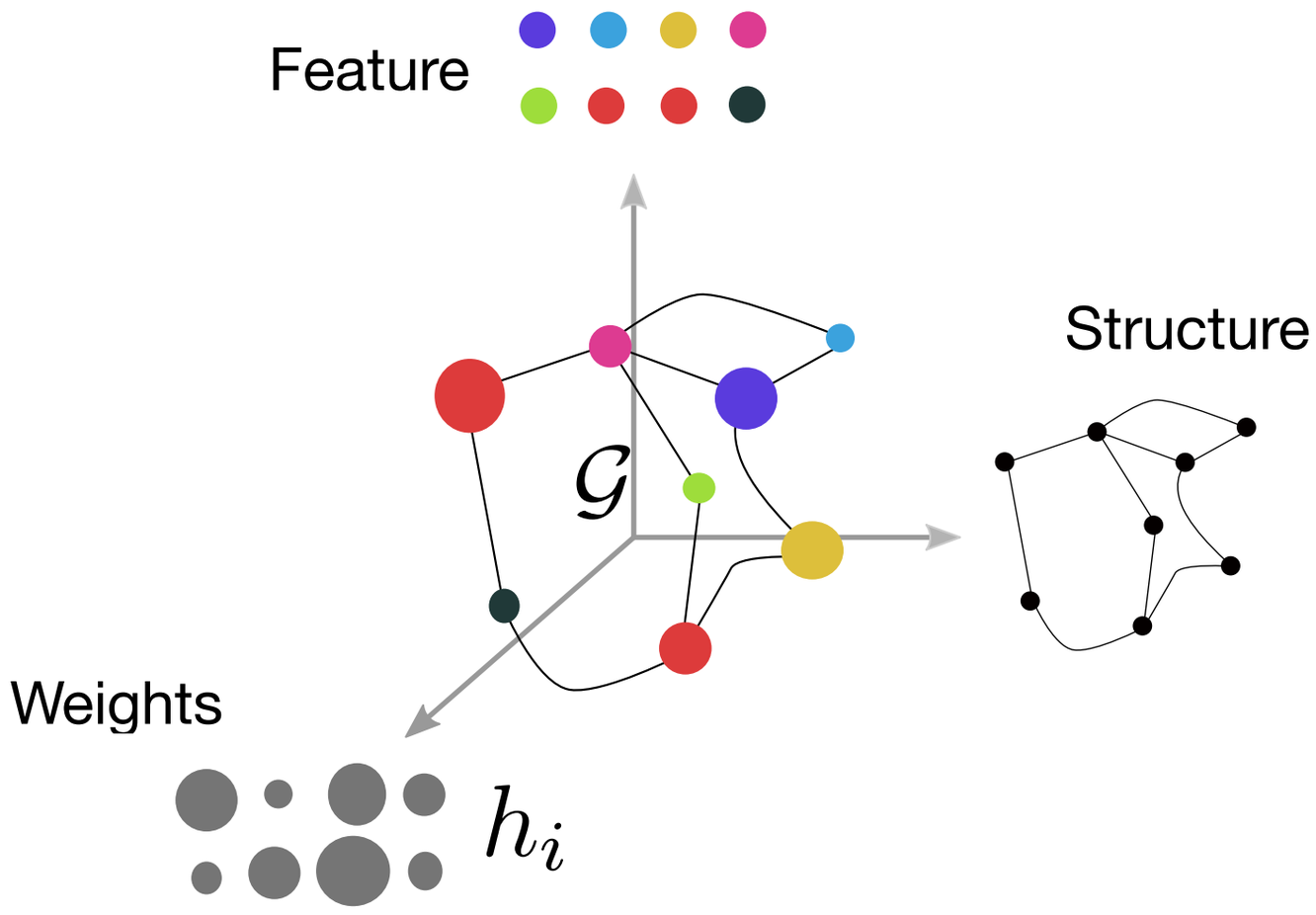
Optimal Transport theory



Use + Develop the Optimal transport theory in this challenging scenario

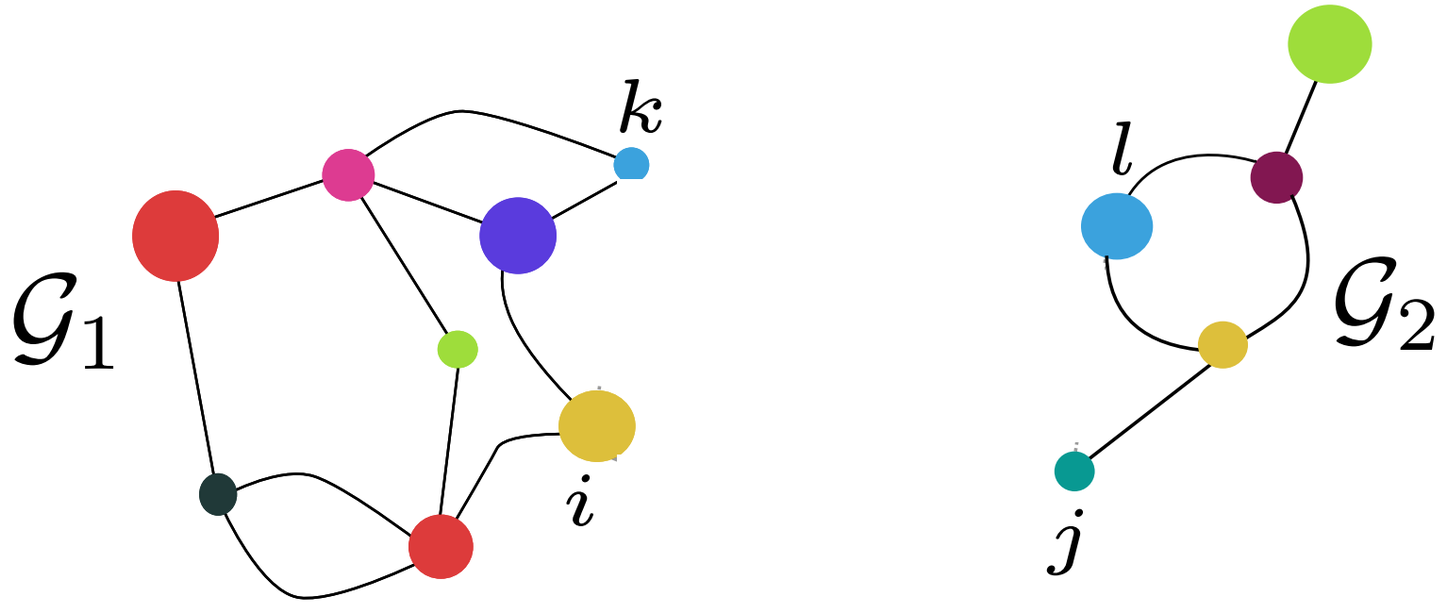
- **Applicability**
- **Mathematical foundations**

Motivation: structured data/ graphs



Motivation: structured data/ graphs

Particularly challenging: dataset with multiple graphs



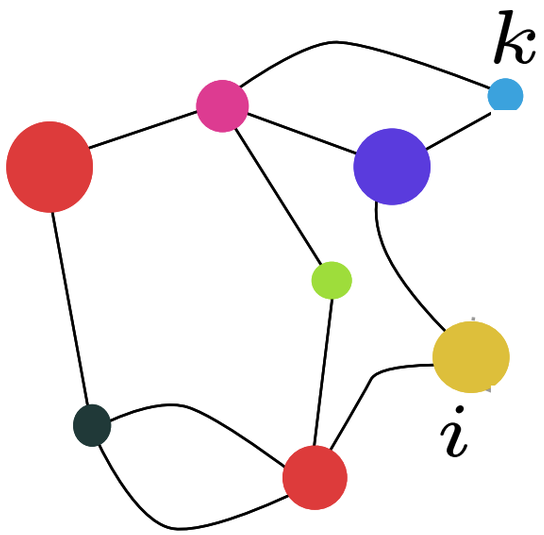
Some questions:

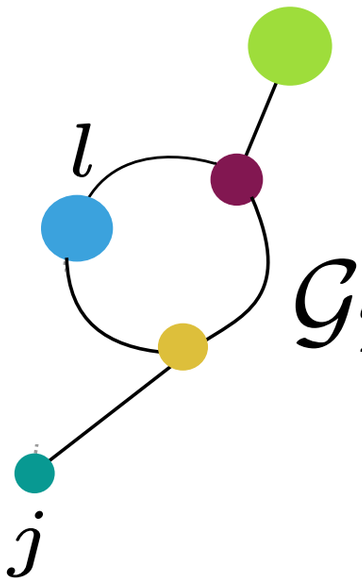
- How to compare the graphs?
- How to operate on them?

Motivation: structured data/ graphs

Particularly challenging: dataset with multiple graphs

$$D\left(\mathcal{G}_1 \quad , \quad \mathcal{G}_2\right) = 12$$





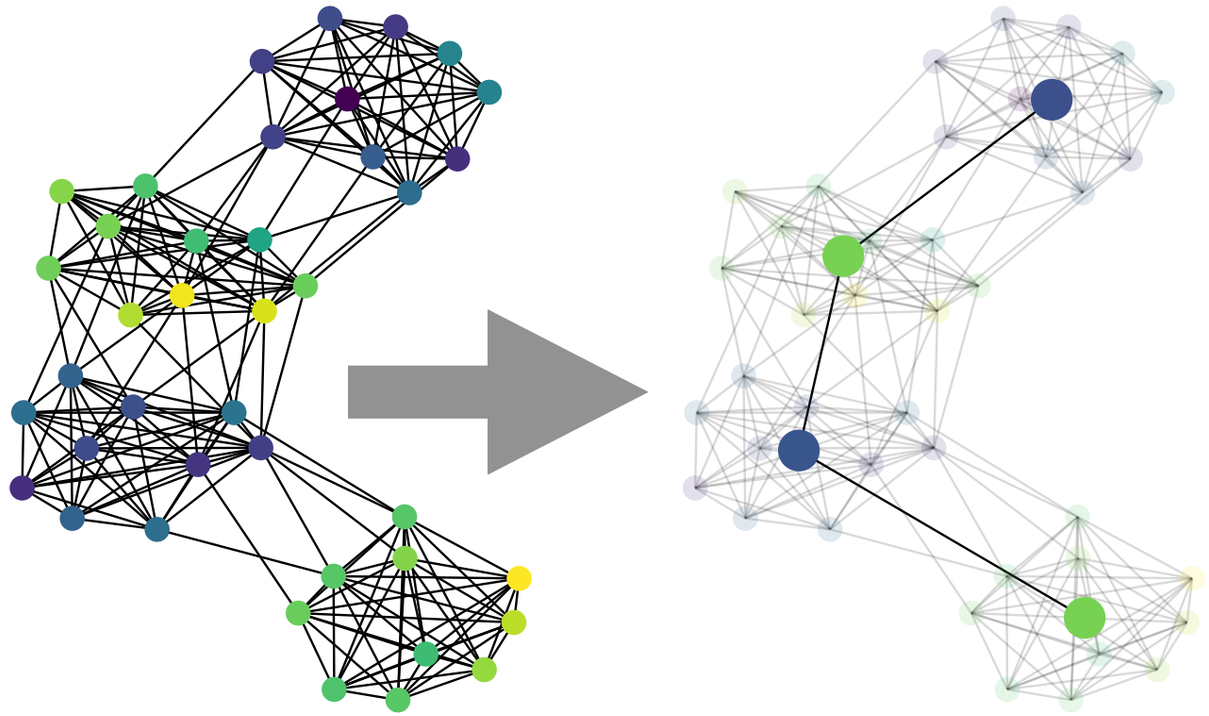
Some questions:

How to compare the graphs?

How to operate on them?

Motivation: structured data/ graphs

Particularly challenging: dataset with multiple graphs

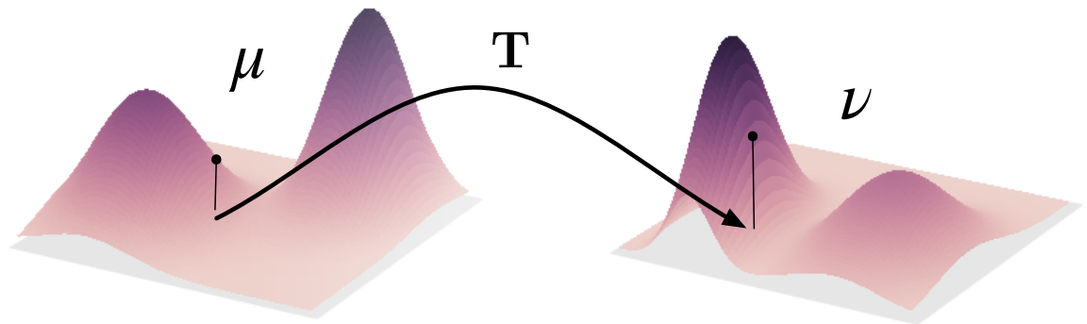


Some questions:

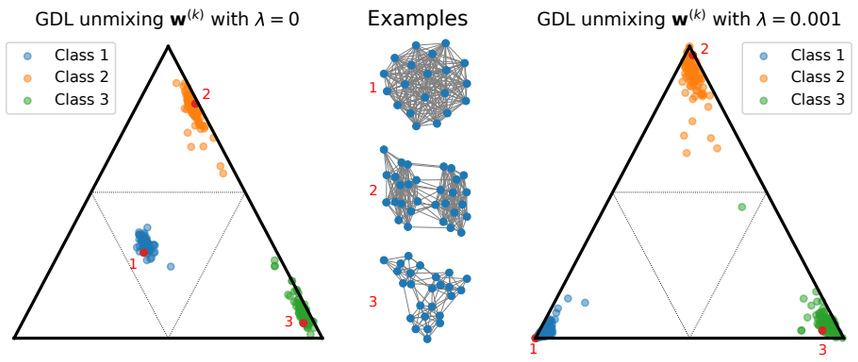
How to compare the graphs?

How to operate on them?

Part I: Optimal Transport

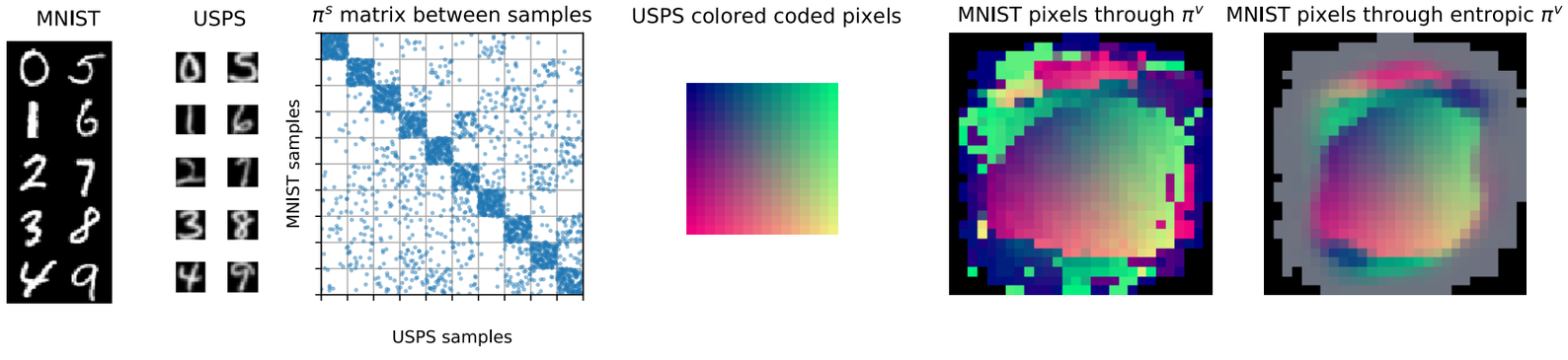


Part II: Online Graph Dictionary Learning



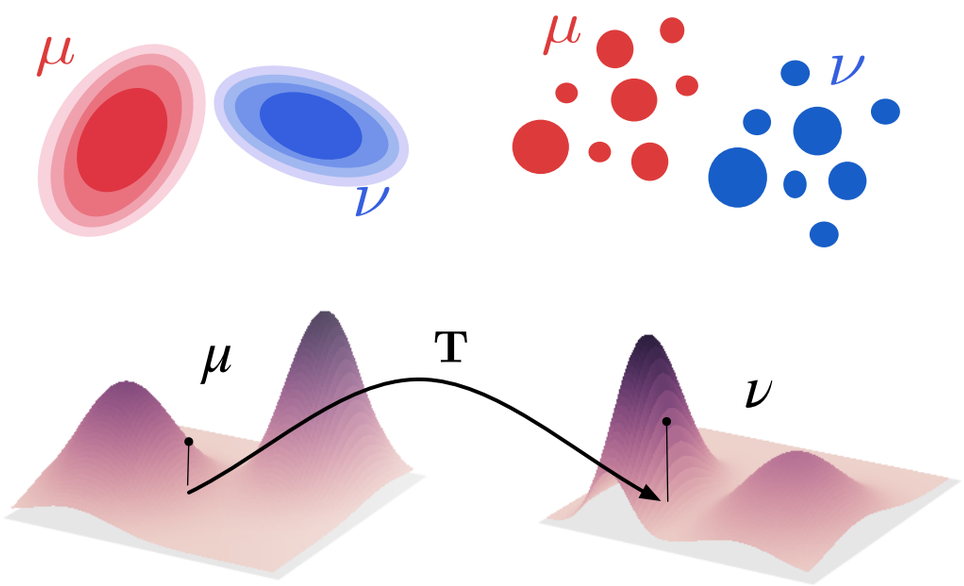
ICML' 2021

Part III: CO-Optimal Transport



NeurIPS' 2020

From linear Optimal Transport to Gromov-Wasserstein



From linear Optimal Transport…

What is it?

Input:

$\mu \in \mathcal{P}(\mathcal{X})$, $\nu \in \mathcal{P}(\mathcal{Y})$

Two probability distributions

Output:

Geometric notion of distance between these distributions

Find correspondences/relations between the samples

From linear Optimal Transport…

Why do we care about probability distributions?

> Measure and probability distributions are at the core of Machine learning

From linear Optimal Transport…

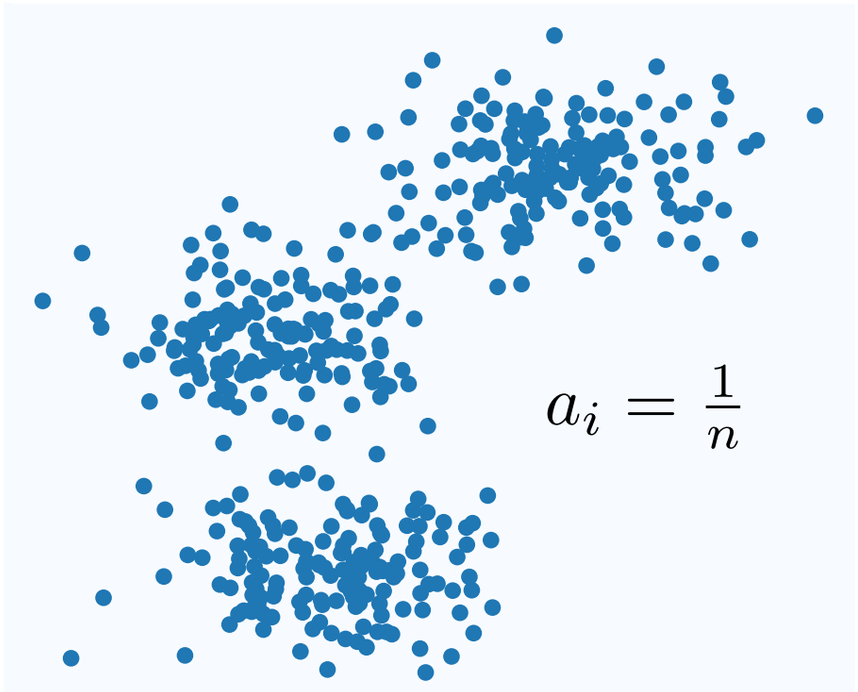
Why do we care about probability distributions?

Measure and probability distributions are at the core of Machine learning

A point of view on the data

Data: $(\mathbf{x}_i)_{i\in[\![n]\!]}$; $\mathbf{x}_i \in \mathbb{R}^d$ ⟶ A probability distribution describing the data

Lagrangian: $\sum_{i=1}^n a_i \delta_{x_i}$



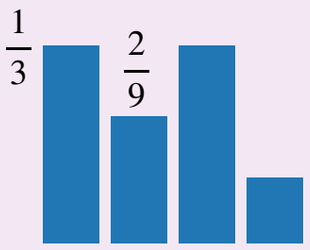
(point clouds)

$\delta_{\mathbf{x}_i}(\mathbf{x}) = 1$ if $\mathbf{x} = \mathbf{x}_i$ else 0

Probability simplex

$\mathbf{a} = (a_i)_{i\in[\![n]\!]} \in \Sigma_n$

$a_i \geq 0, \sum_{i=1}^n a_i = 1$



From linear Optimal Transport…

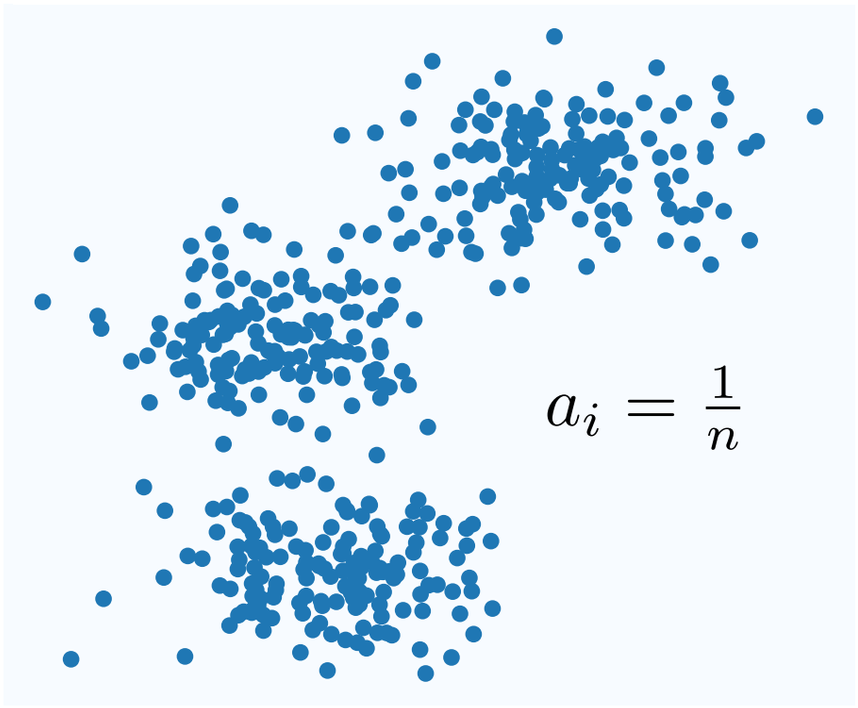
Why do we care about probability distributions?

Measure and probability distributions are at the core of Machine learning

A point of view on the data

Data: $(\mathbf{x}_i)_{i\in[\![n]\!]}$; $\mathbf{x}_i \in \mathbb{R}^d$ ⟶ A probability distribution describing the data

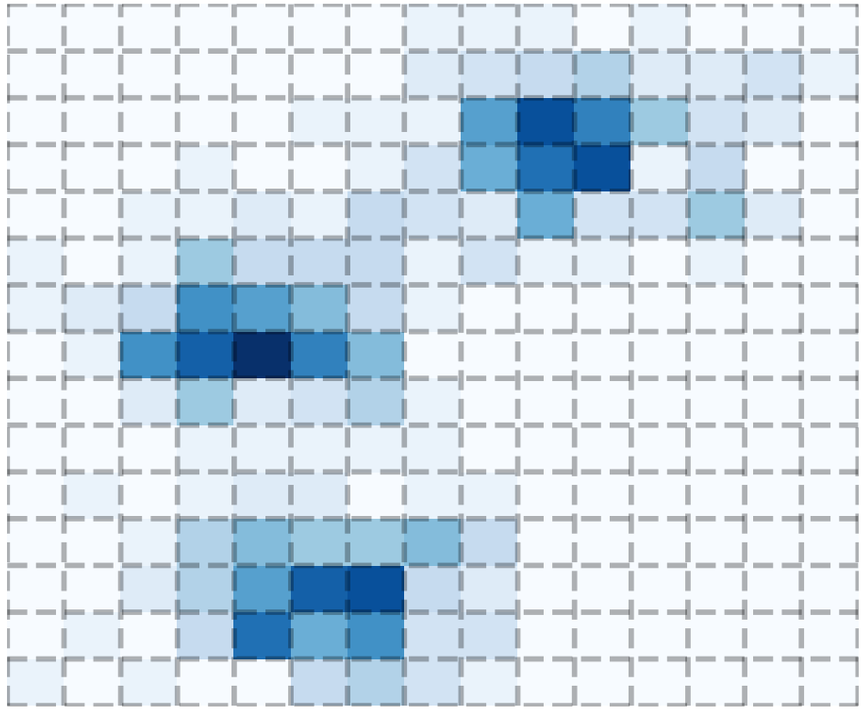
Lagrangian: $\sum_{i=1}^n a_i \delta_{x_i}$



(point clouds)

$\delta_{\mathbf{x}_i}(\mathbf{x}) = 1$ if $\mathbf{x} = \mathbf{x}_i$ else 0

Eulerian: $\sum_{i=1}^N a_i \delta_{\hat{x}_i}$



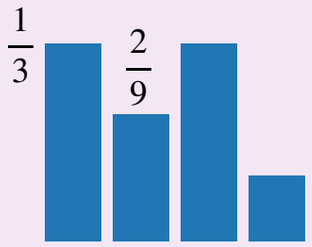
(histograms)

$\hat{x}_i$ fixed position (grid)

Probability simplex

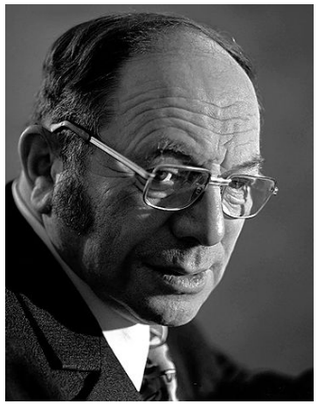
$\mathbf{a} = (a_i)_{i\in[\![n]\!]} \in \Sigma_n$

$a_i \geq 0, \sum_{i=1}^n a_i = 1$



From linear Optimal Transport…

Formulation



Two probability distributions

$$\mu \in \mathcal{P}(\mathcal{X}),\ \nu \in \mathcal{P}(\mathcal{Y})$$

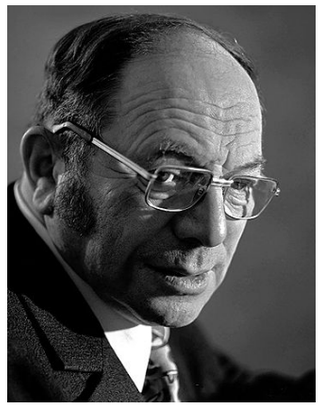
A cost function

$$c(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$$

Optimal Transport

From linear Optimal Transport…

Kantorovitch Formulation



Two probability distributions

$$\mu \in \mathcal{P}(\mathcal{X}),\ \nu \in \mathcal{P}(\mathcal{Y})$$

A cost function

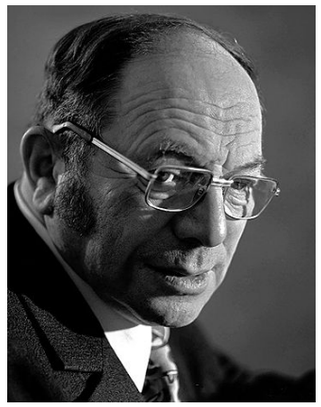
$$c(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$$

Optimal Transport

All the mass of μ is **transported** to ν by a **transport plan** $\pi \in \mathcal{P}(\mathcal{X} \times \mathcal{Y})$

From linear Optimal Transport…

Kantorovitch Formulation



Two probability distributions

$$\mu \in \mathcal{P}(\mathcal{X}),\ \nu \in \mathcal{P}(\mathcal{Y})$$

A cost function

$$c(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$$

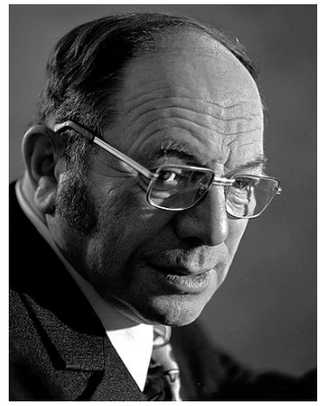
Optimal Transport

All the mass of μ is **transported** to ν by a **transport plan** $\pi \in \mathcal{P}(\mathcal{X} \times \mathcal{Y})$

We want to find the plan that **minimizes the overall cost** of moving all the points

From linear Optimal Transport…

Kantorovitch Formulation: an example



Two probability distributions

$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i} \quad \nu = \sum_{i=j}^{m} b_j \delta_{y_j}$$

A cost function

$$c(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$$

Bakeries = quantity of breads

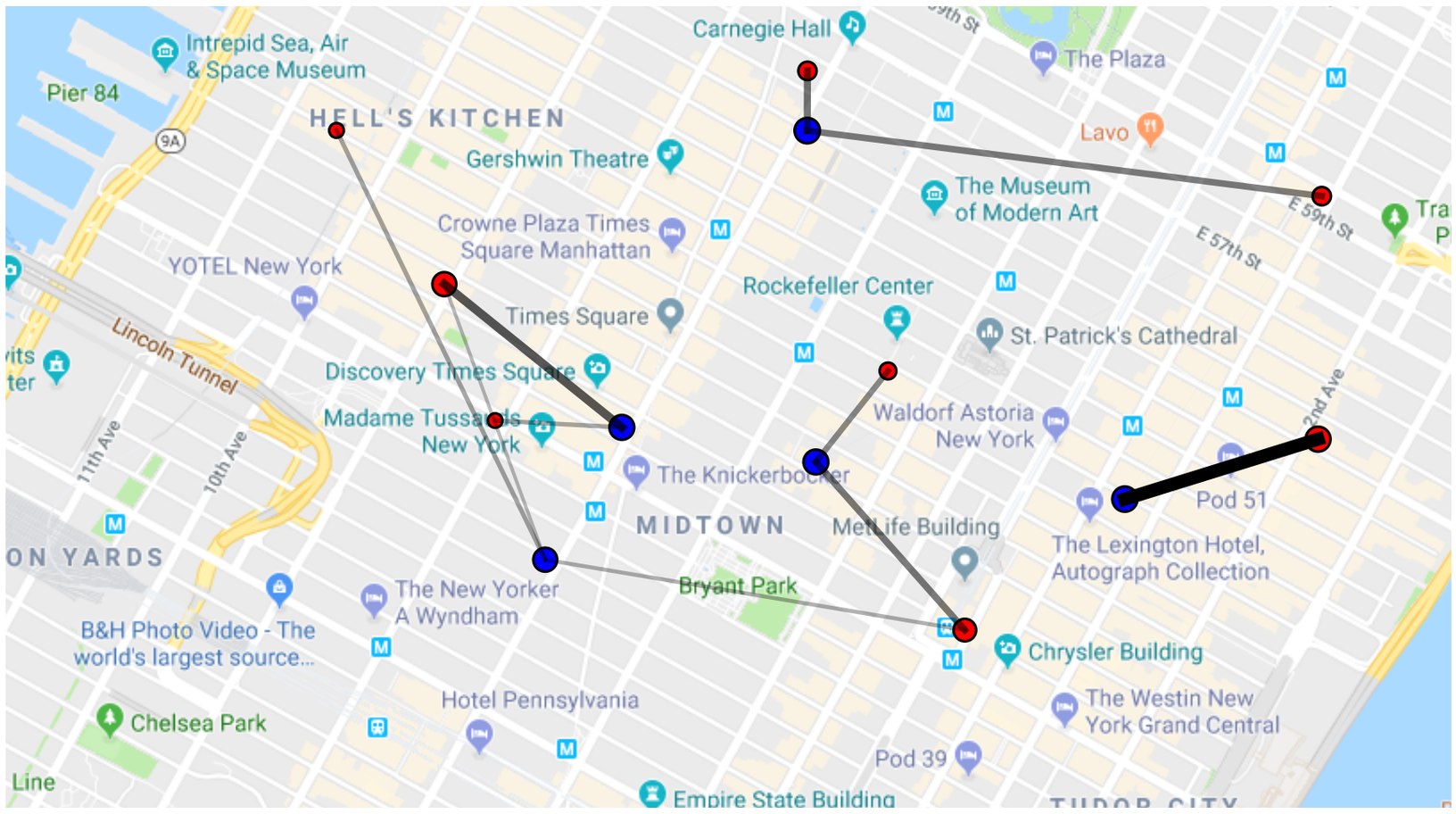
loc: x_i quantity: a_i

Cafés = demand of breads

loc: y_j demand: b_j

Distance between bakeries and cafés

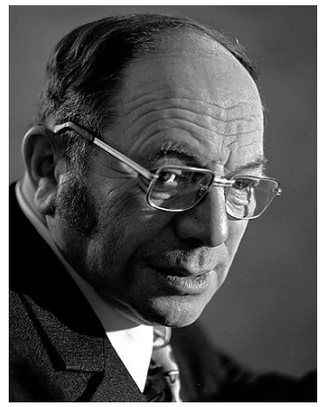
$$c(x_i, y_j)$$



We want to route all the breads from bakeries to cafés the cheapest way

From linear Optimal Transport…

Kantorovitch Formulation: an example



Two probability distributions

$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i} \quad \nu = \sum_{i=j}^{m} b_j \delta_{y_j}$$

A cost function

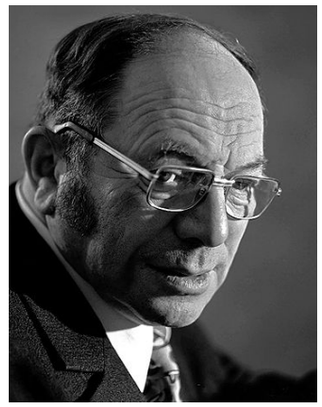
$$c(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$$

Kantorovitch formulation

$$\min_{\boldsymbol{\pi} \in \Pi(\mathbf{a}, \mathbf{b})} \sum_{i,j=1}^{m,n} c(x_i, y_j)\pi_{ij}$$

From linear Optimal Transport…

Kantorovitch Formulation: an example



Two probability distributions

$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i} \quad \nu = \sum_{i=j}^{m} b_j \delta_{y_j}$$

A cost function

$$c(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$$

Kantorovitch formulation

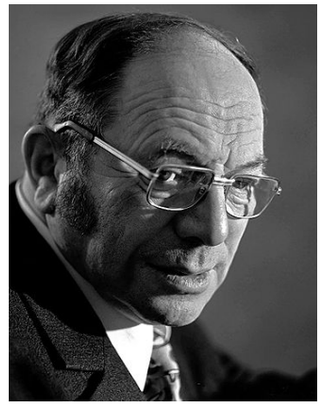
$$\min_{\boldsymbol{\pi} \in \Pi(\mathbf{a}, \mathbf{b})} \sum_{i,j=1}^{m,n} c(x_i, y_j)\pi_{ij}$$

Set of couplings/ transport plans

$$\Pi(\mathbf{a}, \mathbf{b})$$

From linear Optimal Transport…

Kantorovitch Formulation: an example



Two probability distributions

$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i} \quad \nu = \sum_{i=j}^{m} b_j \delta_{y_j}$$

A cost function

$$c(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$$

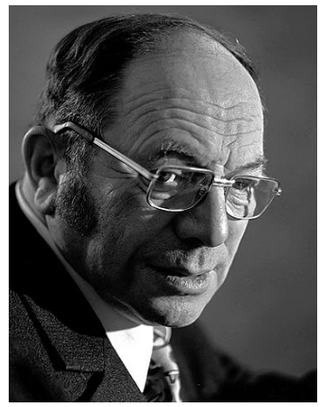
Kantorovitch formulation

$$\min_{\boldsymbol{\pi} \in \Pi(\mathbf{a}, \mathbf{b})} \sum_{i,j=1}^{m,n} c(x_i, y_j) \pi_{ij}$$

How much is shifted from x_i to y_j

From linear Optimal Transport…

Kantorovitch Formulation: an example



Two probability distributions

$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i} \quad \nu = \sum_{i=j}^{m} b_j \delta_{y_j}$$

A cost function

$$c(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$$

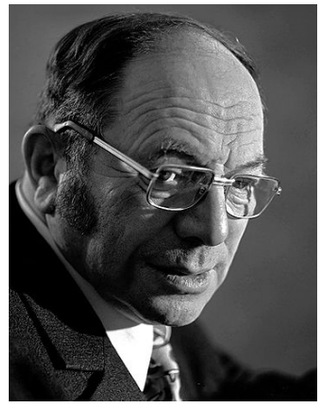
Kantorovitch formulation

$$\min_{\boldsymbol{\pi} \in \Pi(\mathbf{a},\mathbf{b})} \sum_{i,j=1}^{m,n} c(x_i, y_j)\pi_{ij}$$

Cost of moving masses from x_i to y_j

From linear Optimal Transport…

Kantorovitch Formulation: an example



Two probability distributions

$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i} \quad \nu = \sum_{i=j}^{m} b_j \delta_{y_j}$$

A cost function

$$c(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$$

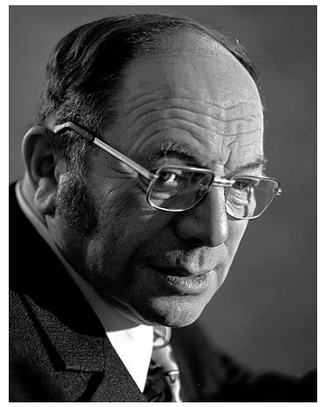
Kantorovitch formulation

$$\min_{\boldsymbol{\pi} \in \Pi(\mathbf{a}, \mathbf{b})} \sum_{i,j=1}^{m,n} c(x_i, y_j)\pi_{ij}$$

Total cost

From linear Optimal Transport…

Kantorovitch Formulation: an example



Two probability distributions

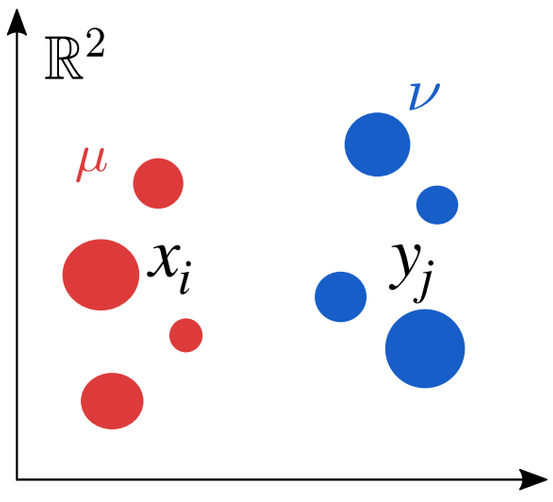
$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i} \quad \nu = \sum_{i=j}^{m} b_j \delta_{y_j}$$

A cost function

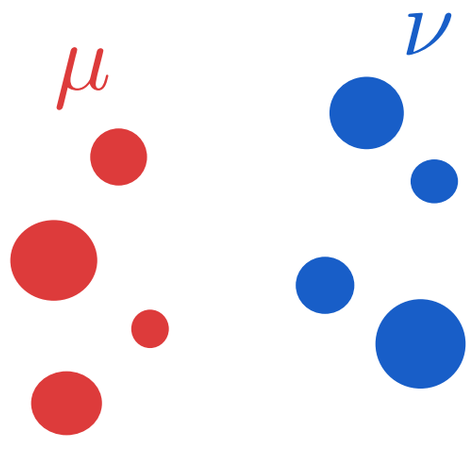
$$c(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$$

Kantorovitch formulation

$$\min_{\boldsymbol{\pi} \in \Pi(\mathbf{a},\mathbf{b})} \sum_{i,j=1}^{m,n} c(x_i, y_j)\pi_{ij}$$

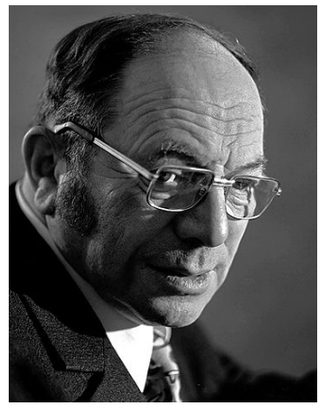


$$\Pi(\mathbf{a}, \mathbf{b}) = \{\boldsymbol{\pi} \in \mathbb{R}_+^{n \times m} \mid \forall (i,j), \sum_{j=1}^{m} \pi_{ij} = a_i, \ \sum_{i=1}^{n} \pi_{ij} = b_j\}$$



From linear Optimal Transport…

Kantorovitch Formulation: an example



Two probability distributions

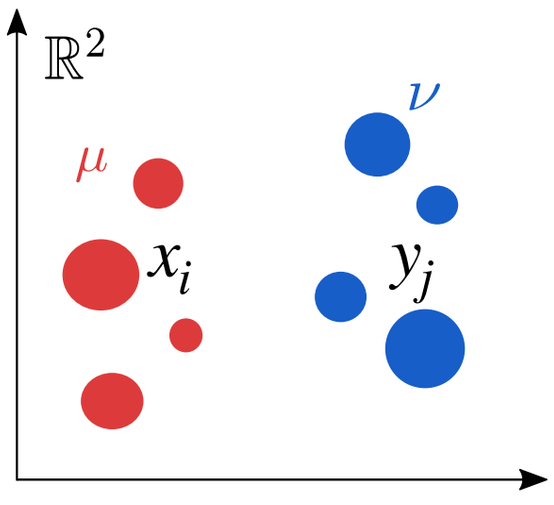
$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i} \quad \nu = \sum_{i=j}^{m} b_j \delta_{y_j}$$

A cost function

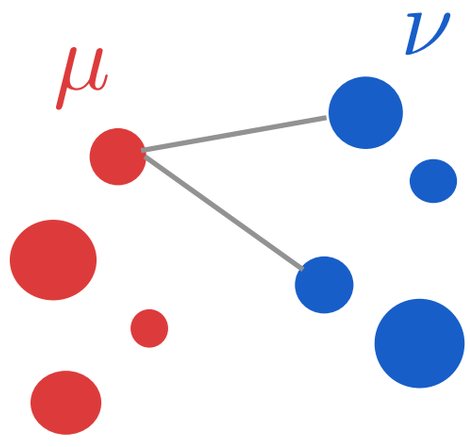
$$c(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$$

Kantorovitch formulation

$$\min_{\boldsymbol{\pi} \in \Pi(\mathbf{a},\mathbf{b})} \sum_{i,j=1}^{m,n} c(x_i, y_j)\pi_{ij}$$

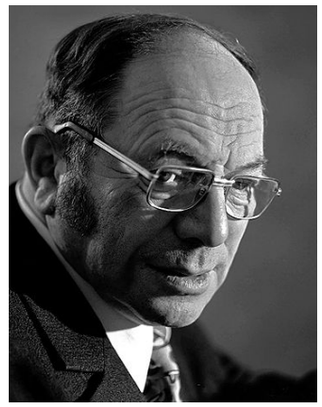


$$\Pi(\mathbf{a}, \mathbf{b}) = \{\boldsymbol{\pi} \in \mathbb{R}_+^{n \times m} \mid \forall (i,j), \sum_{j=1}^{m} \pi_{ij} = a_i, \ \sum_{i=1}^{n} \pi_{ij} = b_j\}$$



From linear Optimal Transport…

Kantorovitch Formulation: an example



Two probability distributions

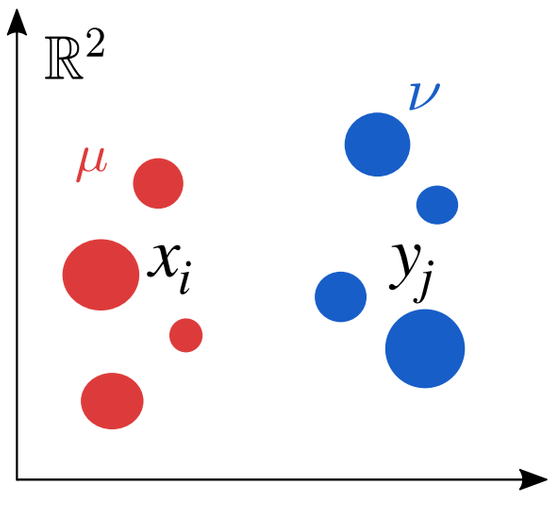
$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i} \quad \nu = \sum_{i=j}^{m} b_j \delta_{y_j}$$

A cost function

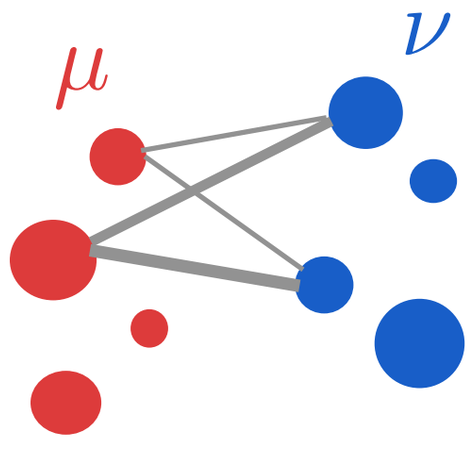
$$c(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$$

Kantorovitch formulation

$$\min_{\boldsymbol{\pi} \in \Pi(\mathbf{a},\mathbf{b})} \sum_{i,j=1}^{m,n} c(x_i, y_j)\pi_{ij}$$

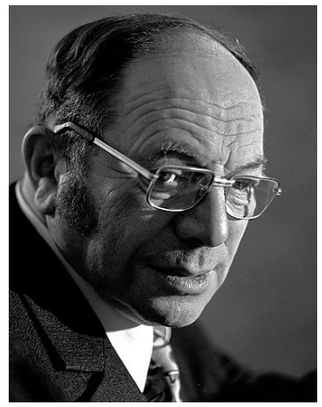


$$\Pi(\mathbf{a}, \mathbf{b}) = \{\boldsymbol{\pi} \in \mathbb{R}_+^{n \times m} \mid \forall (i,j), \sum_{j=1}^{m} \pi_{ij} = a_i, \ \sum_{i=1}^{n} \pi_{ij} = b_j\}$$



From linear Optimal Transport…

Kantorovitch Formulation: an example



Two probability distributions

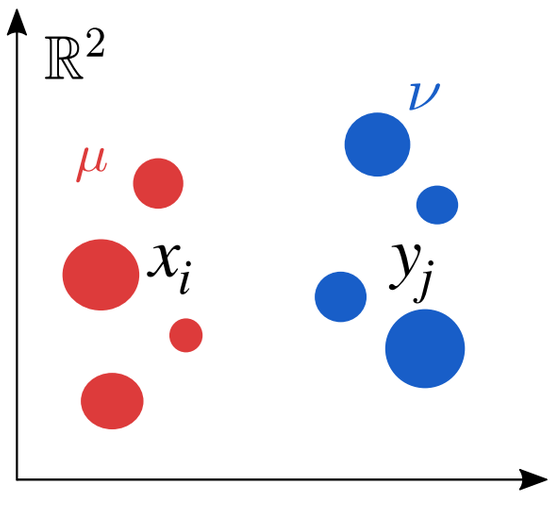
$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i} \quad \nu = \sum_{i=j}^{m} b_j \delta_{y_j}$$

A cost function

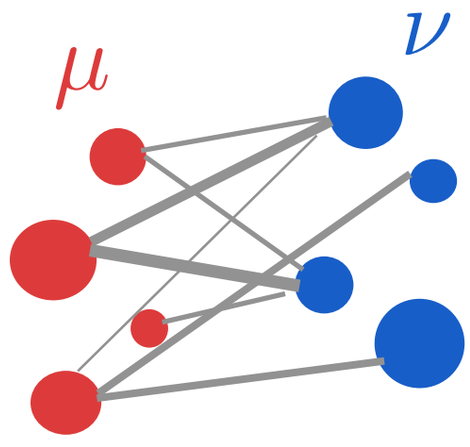
$$c(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$$

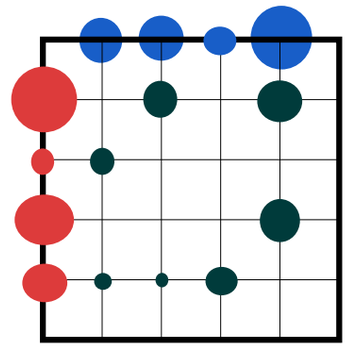
Kantorovitch formulation

$$\min_{\boldsymbol{\pi} \in \Pi(\mathbf{a},\mathbf{b})} \sum_{i,j=1}^{m,n} c(x_i, y_j)\pi_{ij}$$



$$\Pi(\mathbf{a}, \mathbf{b}) = \{\boldsymbol{\pi} \in \mathbb{R}_+^{n \times m} \mid \forall (i,j), \sum_{j=1}^{m} \pi_{ij} = a_i, \ \sum_{i=1}^{n} \pi_{ij} = b_j\}$$

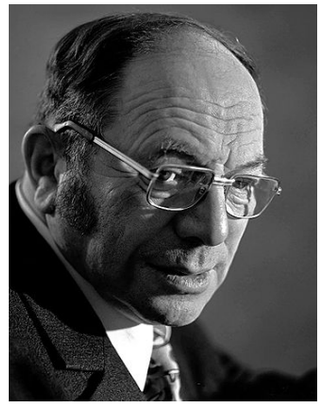




$\pi \in \mathbb{R}_+^{n \times m}$

From linear Optimal Transport...

Kantorovitch Formulation: general case



Two probability distributions

$$\mu \in \mathcal{P}(\mathcal{X}),\ \nu \in \mathcal{P}(\mathcal{Y})$$

A cost function

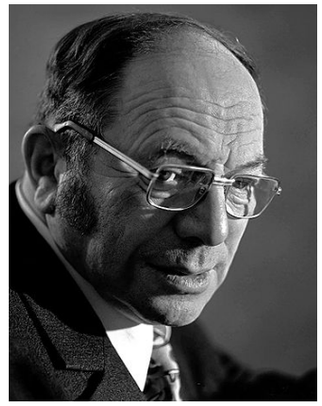
$$c(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$$

Kantorovitch formulation

$$\mathcal{T}_c(\mu, \nu) = \inf_{\pi \in \Pi(\mu, \nu)} \int_{\mathcal{X} \times \mathcal{Y}} c(x, y) \mathrm{d}\pi(x, y)$$

From linear Optimal Transport…

Wasserstein distance



Two probability distributions

$$\mu \in \mathcal{P}(\Omega), \nu \in \mathcal{P}(\Omega)$$

A distance

$$d : \Omega \times \Omega \to \mathbb{R}_+$$

Example: $\Omega = \mathbb{R}^d$

Wasserstein distance

$$W_p^p(\mu, \nu) = \inf_{\pi \in \Pi(\mu,\nu)} \int_{\Omega \times \Omega} d^p(x, y) \mathrm{d}\pi(x, y)$$

Result:

$\mathcal{P}(\Omega)$ is a metric space

$$W_p(\mu, \nu) = 0 \iff \mu = \nu$$

...to Gromov-Wasserstein

What if ?

Data are in Incomparable spaces

Two probability distributions

$$\mu \in \mathcal{P}(\mathcal{X}), \nu \in \mathcal{P}(\mathcal{Y}) \text{ with } \mathcal{X}, \mathcal{Y} \nsubseteq \Omega$$

A cost function ?????

$$c(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$$

⇨ Not straightforward to find a suitable cost (e.g. no distance available)

...to Gromov-Wasserstein

What if ?

Data are in Incomparable spaces

Two probability distributions

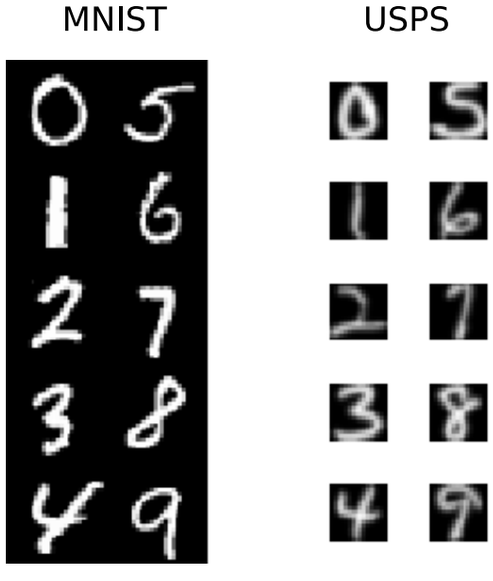
$$\mu \in \mathcal{P}(\mathcal{X}), \nu \in \mathcal{P}(\mathcal{Y}) \text{ with } \mathcal{X}, \mathcal{Y} \nsubseteq \Omega$$

A cost function ?????

$$c(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$$

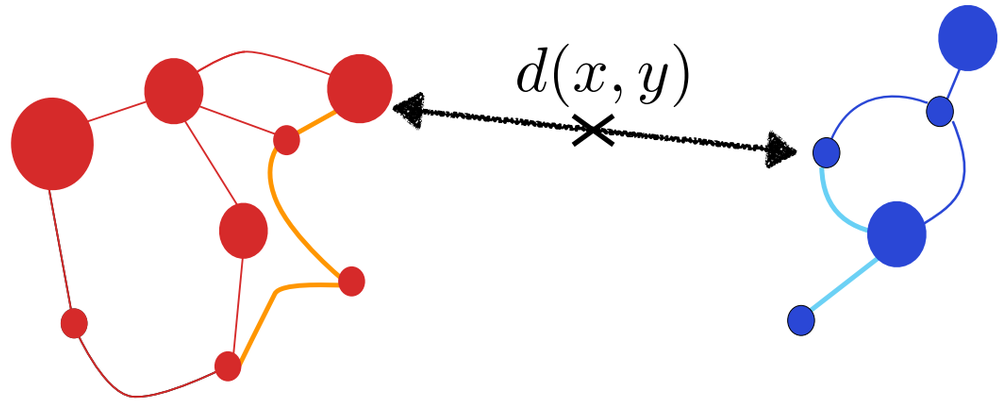
⇨ Not straightforward to find a suitable cost (e.g. no distance available)

Different Euclidean spaces



Example: $\mathcal{X} = \mathbb{R}^{28*28}, \mathcal{Y} = \mathbb{R}^{16*16}$

Samples = nodes of different graphs



Example: $\mathcal{X} = \text{Graph } 1, \mathcal{Y} = \text{Graph } 2$

…to Gromov-Wasserstein

Gromov-Wasserstein distance

Two probability distributions

$$\mu \in \mathcal{P}(\mathcal{X}), \nu \in \mathcal{P}(\mathcal{Y})$$

Two « intra-domain » costs

$$c_{\mathcal{X}} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$
$$c_{\mathcal{Y}} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$$

Gromov-Wasserstein distance

$$GW_p^p(c_{\mathcal{X}}, c_{\mathcal{Y}}, \mu, \nu) = \inf_{\pi \in \Pi(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{Y}} \int_{\mathcal{X}\times\mathcal{Y}} \left|c_{\mathcal{X}}(x,x') - c_{\mathcal{Y}}(y,y')\right|^p \mathrm{d}\pi(x,y)\mathrm{d}\pi(x',y'))$$

...to Gromov-Wasserstein

Gromov-Wasserstein distance

Two probability distributions

$$\mu \in \mathcal{P}(\mathcal{X}), \nu \in \mathcal{P}(\mathcal{Y})$$

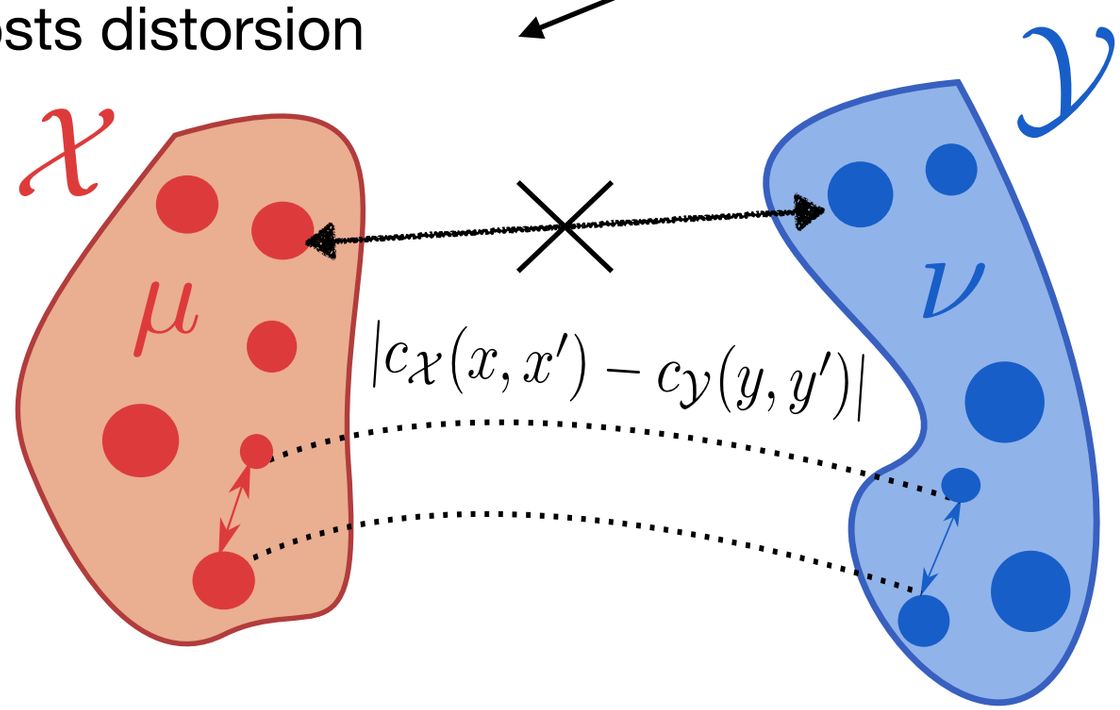
Two « intra-domain » costs

$$c_{\mathcal{X}} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$
$$c_{\mathcal{Y}} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$$

Gromov-Wasserstein distance

$$GW_p^p(c_{\mathcal{X}}, c_{\mathcal{Y}}, \mu, \nu) = \inf_{\pi \in \Pi(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{Y}} \int_{\mathcal{X}\times\mathcal{Y}} \left|c_{\mathcal{X}}(x,x') - c_{\mathcal{Y}}(y,y')\right|^p \mathrm{d}\pi(x,y)\mathrm{d}\pi(x',y'))$$

Measure the costs distorsion



...to Gromov-Wasserstein

Gromov-Wasserstein distance

Two probability distributions

$$\mu \in \mathcal{P}(\mathcal{X}), \nu \in \mathcal{P}(\mathcal{Y})$$

Two « intra-domain » costs

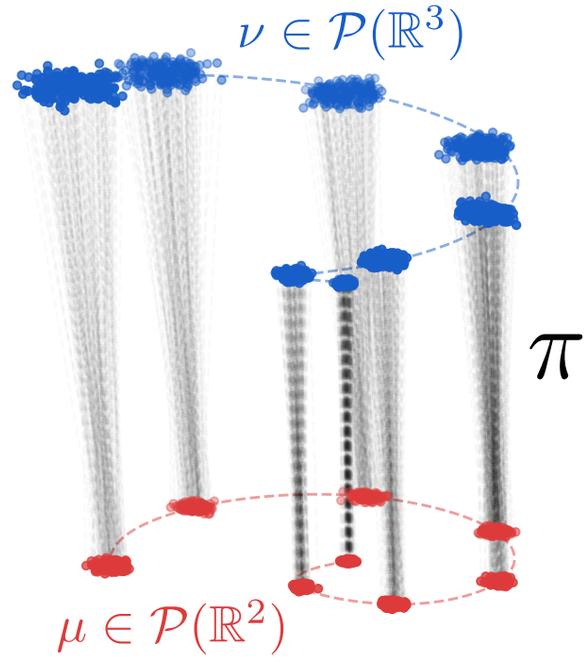
$$c_{\mathcal{X}} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$
$$c_{\mathcal{Y}} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$$

Gromov-Wasserstein distance

$$GW_p^p(c_{\mathcal{X}}, c_{\mathcal{Y}}, \mu, \nu) = \inf_{\pi \in \Pi(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{Y}} \int_{\mathcal{X}\times\mathcal{Y}} \left|c_{\mathcal{X}}(x,x') - c_{\mathcal{Y}}(y,y')\right|^p \mathrm{d}\pi(x,y)\mathrm{d}\pi(x',y'))$$

The transportation problem is not linear anymore but **quadratic**

Associate pair of points with similar costs in each space



...to Gromov-Wasserstein

Gromov-Wasserstein distance


Gromov-Wasserstein distance

$$GW_p^p(c_{\mathcal{X}}, c_{\mathcal{Y}}, \mu, \nu) = \inf_{\pi \in \Pi(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{Y}} \int_{\mathcal{X}\times\mathcal{Y}} \left|c_{\mathcal{X}}(x,x') - c_{\mathcal{Y}}(y,y')\right|^p \mathrm{d}\pi(x,y)\mathrm{d}\pi(x',y'))$$

A distance w.r.t isomorphism

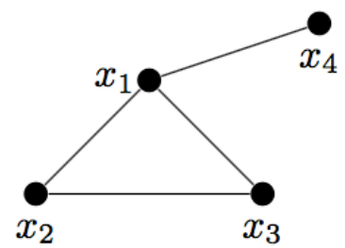
GW is a distance on the "space of all spaces":

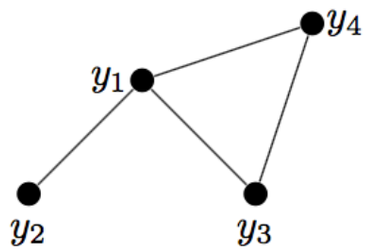
$$\mathbb{X} = \{(\mathcal{X}, d_{\mathcal{X}}, \mu \in \mathcal{P}(\mathcal{X})); d_{\mathcal{X}} \text{ metric } \} \text{ (mm-spaces)}$$

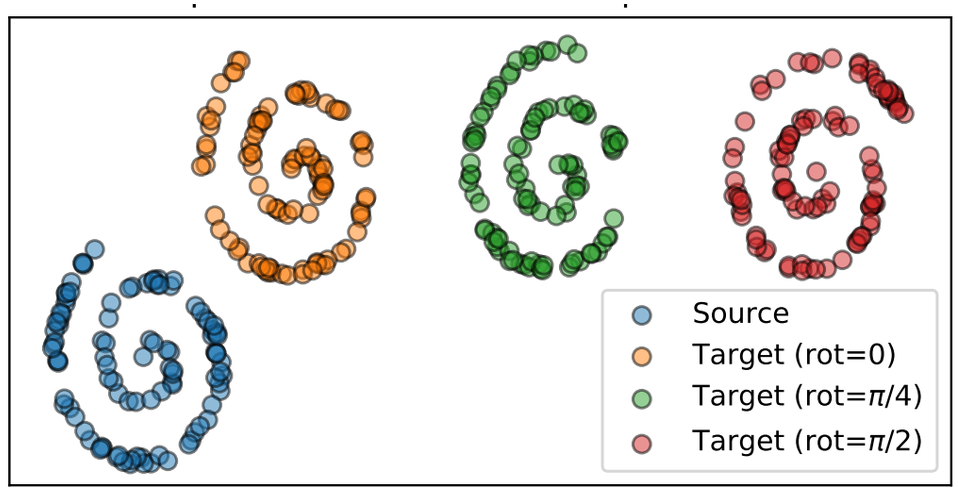
- $GW_p(d_{\mathcal{X}}, d_{\mathcal{Y}}, \mu, \nu) = 0$ iff $\exists \phi : \mathcal{X} \to \mathcal{Y}$

  ϕ is a isometry $d_{\mathcal{X}}(x, x') = d_{\mathcal{Y}}(\phi(x), \phi(x'))$

Isometry: permutations, rotations, translations,...







...to Gromov-Wasserstein

Gromov-Wasserstein distance


Gromov-Wasserstein distance

$$GW_p^p(c_{\mathcal{X}}, c_{\mathcal{Y}}, \mu, \nu) = \inf_{\pi \in \Pi(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{Y}} \int_{\mathcal{X}\times\mathcal{Y}} \left|c_{\mathcal{X}}(x,x') - c_{\mathcal{Y}}(y,y')\right|^p \mathrm{d}\pi(x,y)\mathrm{d}\pi(x',y'))$$

A distance w.r.t isomorphism

GW is a distance on the "space of all spaces":

$\mathbb{X} = \{(\mathcal{X}, d_{\mathcal{X}}, \mu \in \mathcal{P}(\mathcal{X})); d_{\mathcal{X}} \text{ metric } \}$ (mm-spaces)

- $GW_p(d_{\mathcal{X}}, d_{\mathcal{Y}}, \mu, \nu) = 0$ iff $\exists \phi : \mathcal{X} \to \mathcal{Y}$

  ϕ is a isometry $d_{\mathcal{X}}(x,x') = d_{\mathcal{Y}}(\phi(x), \phi(x'))$

  ϕ is measure-preserving: $\phi \# \mu = \nu$

Push-forward $\phi \# \mu$

$$\mu = \sum_{i=1}^n a_i \delta_{x_i} \underset{\phi\#\mu}{\to} \sum_{i=1}^n a_i \delta_{\phi(x_i)}$$

...to Gromov-Wasserstein

Gromov-Wasserstein distance


Gromov-Wasserstein distance

$$GW_p^p(c_{\mathcal{X}}, c_{\mathcal{Y}}, \mu, \nu) = \inf_{\pi \in \Pi(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{Y}} \int_{\mathcal{X}\times\mathcal{Y}} \left|c_{\mathcal{X}}(x,x') - c_{\mathcal{Y}}(y,y')\right|^p \mathrm{d}\pi(x,y)\mathrm{d}\pi(x',y'))$$

A distance w.r.t isomorphism

GW is a distance on the "space of all spaces":

$$\mathbb{X} = \{(\mathcal{X}, d_{\mathcal{X}}, \mu \in \mathcal{P}(\mathcal{X})); d_{\mathcal{X}} \text{ metric } \} \text{ (mm-spaces)}$$

- $GW_p(d_{\mathcal{X}}, d_{\mathcal{Y}}, \mu, \nu) = 0$ iff $\exists \phi : \mathcal{X} \to \mathcal{Y}$

  ϕ is a isometry $d_{\mathcal{X}}(x,x') = d_{\mathcal{Y}}(\phi(x), \phi(x'))$

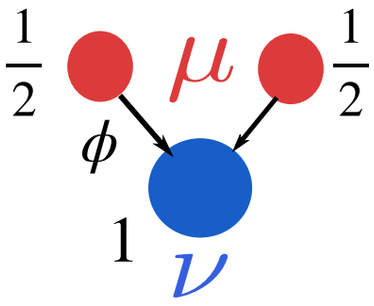
  ϕ is measure-preserving: $\phi \# \mu = \nu$

(Weights are compatible)

Push-forward $\phi \# \mu$

$$\mu = \sum_{i=1}^n a_i \delta_{x_i} \underset{\phi\#\mu}{\to} \sum_{i=1}^n a_i \delta_{\phi(x_i)}$$

Compatible



$$\frac{1}{2} + \frac{1}{2} \to 1$$

...to Gromov-Wasserstein

Gromov-Wasserstein distance


Gromov-Wasserstein distance

$$GW_p^p(c_{\mathcal{X}}, c_{\mathcal{Y}}, \mu, \nu) = \inf_{\pi \in \Pi(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{Y}} \int_{\mathcal{X}\times\mathcal{Y}} \left|c_{\mathcal{X}}(x,x') - c_{\mathcal{Y}}(y,y')\right|^p \mathrm{d}\pi(x,y)\mathrm{d}\pi(x',y'))$$

A distance w.r.t isomorphism

GW is a distance on the "space of all spaces":

$$\mathbb{X} = \{(\mathcal{X}, d_{\mathcal{X}}, \mu \in \mathcal{P}(\mathcal{X})); d_{\mathcal{X}} \text{ metric } \} \text{ (mm-spaces)}$$

- $GW_p(d_{\mathcal{X}}, d_{\mathcal{Y}}, \mu, \nu) = 0$ iff $\exists \phi : \mathcal{X} \to \mathcal{Y}$

  ϕ is a isometry $d_{\mathcal{X}}(x,x') = d_{\mathcal{Y}}(\phi(x), \phi(x'))$

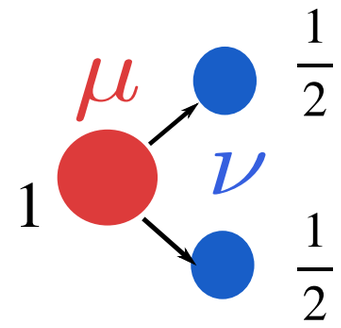
  ϕ is measure-preserving: $\phi \# \mu = \nu$

(Weights are compatible)

Push-forward $\phi \# \mu$

$$\mu = \sum_{i=1}^n a_i \delta_{x_i} \underset{\phi\#\mu}{\to} \sum_{i=1}^n a_i \delta_{\phi(x_i)}$$

Not compatible



$$1 \nrightarrow (\frac{1}{2}, \frac{1}{2})$$

...to Gromov-Wasserstein

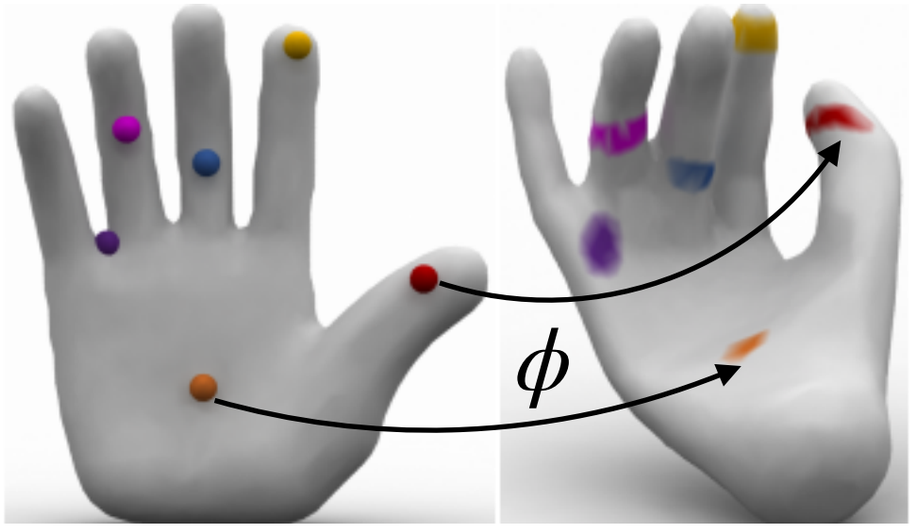
Gromov-Wasserstein distance

Gromov-Wasserstein = a bending invariant distance

- $GW_p(d_{\mathcal{X}}, d_{\mathcal{Y}}, \mu, \nu) = 0$ iff $\exists \phi : \mathcal{X} \to \mathcal{Y}$

  ϕ is a isometry $d_{\mathcal{X}}(x, x') = d_{\mathcal{Y}}(\phi(x), \phi(x'))$

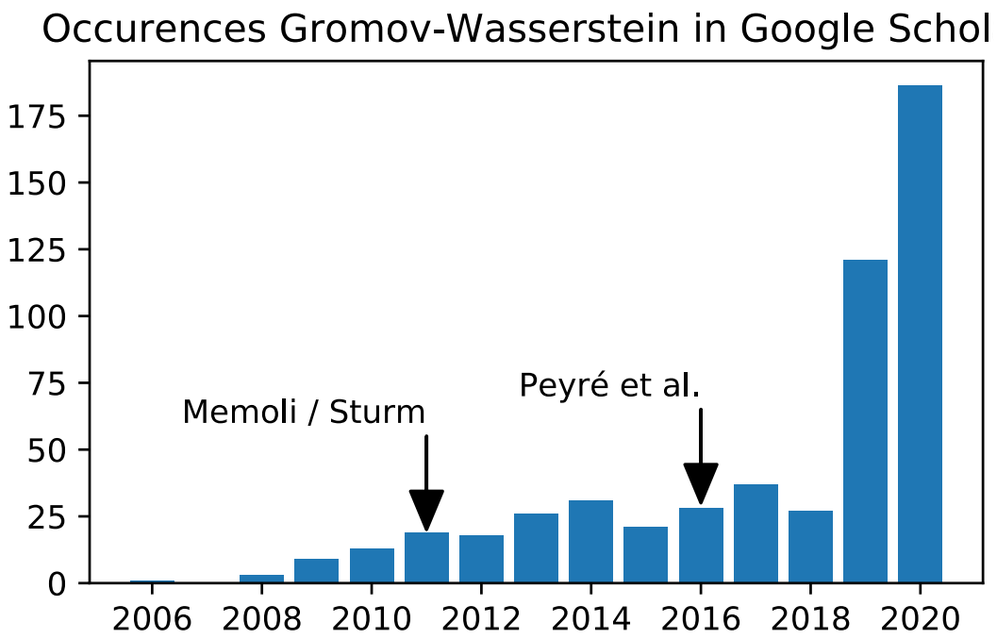
  ϕ is measure-preserving $\phi \# \mu = \nu$



[Solomon 2016]

Applications for geometric data

- Barycenter of relational data [Peyré 2016], Point clouds/meshes [Ezuz 2017]
- Shape comparison [Mémoli 2011, Solomon 2016]
- Graphs [Xu 2019, Fey 2020], biology [Demetci 2020], generative modeling [Bunne 2019]



Solving OT

Solving OT

A linear problem

Discrete probability measures

$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i} \quad \nu = \sum_{j=1}^{m} b_j \delta_{y_j}$$

Linear Program:

$$\min_{\boldsymbol{\pi} \in \Pi(\mathbf{a},\mathbf{b})} \sum_{ij} c_{i,j} \pi_{i,j} = \min_{\boldsymbol{\pi} \in \Pi(\mathbf{a},\mathbf{b})} \langle \mathbf{C}, \boldsymbol{\pi} \rangle$$

Simplex, Network flow, Hungarian algorithms $\sim O(n^3 \log(n))$

Solving OT

A linear problem

Discrete probability measures

$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i} \quad \nu = \sum_{j=1}^{m} b_j \delta_{y_j}$$

Linear Program:

$$\min_{\boldsymbol{\pi} \in \Pi(\mathbf{a},\mathbf{b})} \sum_{ij} c_{i,j} \pi_{i,j} = \min_{\boldsymbol{\pi} \in \Pi(\mathbf{a},\mathbf{b})} \langle \mathbf{C}, \boldsymbol{\pi} \rangle$$

Simplex, Network flow, Hungarian algorithms $\sim O(n^3 \log(n))$

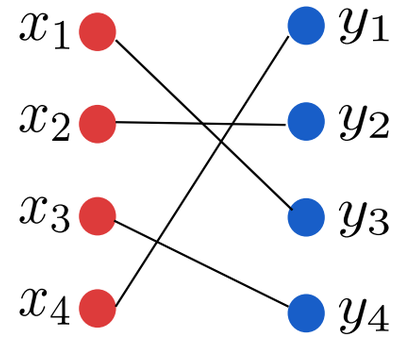
Uniform weights

$$\mathbf{a} = \mathbf{b} = \frac{\mathbf{1}_n}{n}$$

Monge Problem

$$\min_{\sigma \in S_n} \sum_{i=1}^{n} c_{i,\sigma(i)}$$

One-to-one



Solving OT

A linear problem

Discrete probability measures

$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i} \quad \nu = \sum_{j=1}^{m} b_j \delta_{y_j}$$

Linear Program:

$$\min_{\boldsymbol{\pi} \in \Pi(\mathbf{a},\mathbf{b})} \sum_{ij} c_{i,j} \pi_{i,j} = \min_{\boldsymbol{\pi} \in \Pi(\mathbf{a},\mathbf{b})} \langle \mathbf{C}, \boldsymbol{\pi} \rangle$$

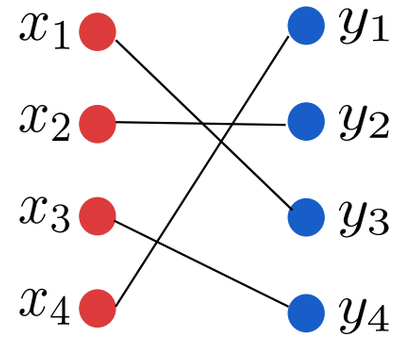
Simplex, Network flow, Hungarian algorithms $\sim O(n^3 \log(n))$

Uniform weights

$$\mathbf{a} = \mathbf{b} = \frac{\mathbf{1}_n}{n}$$

Monge Problem

$$\min_{\sigma \in S_n} \sum_{i=1}^{n} c_{i,\sigma(i)}$$



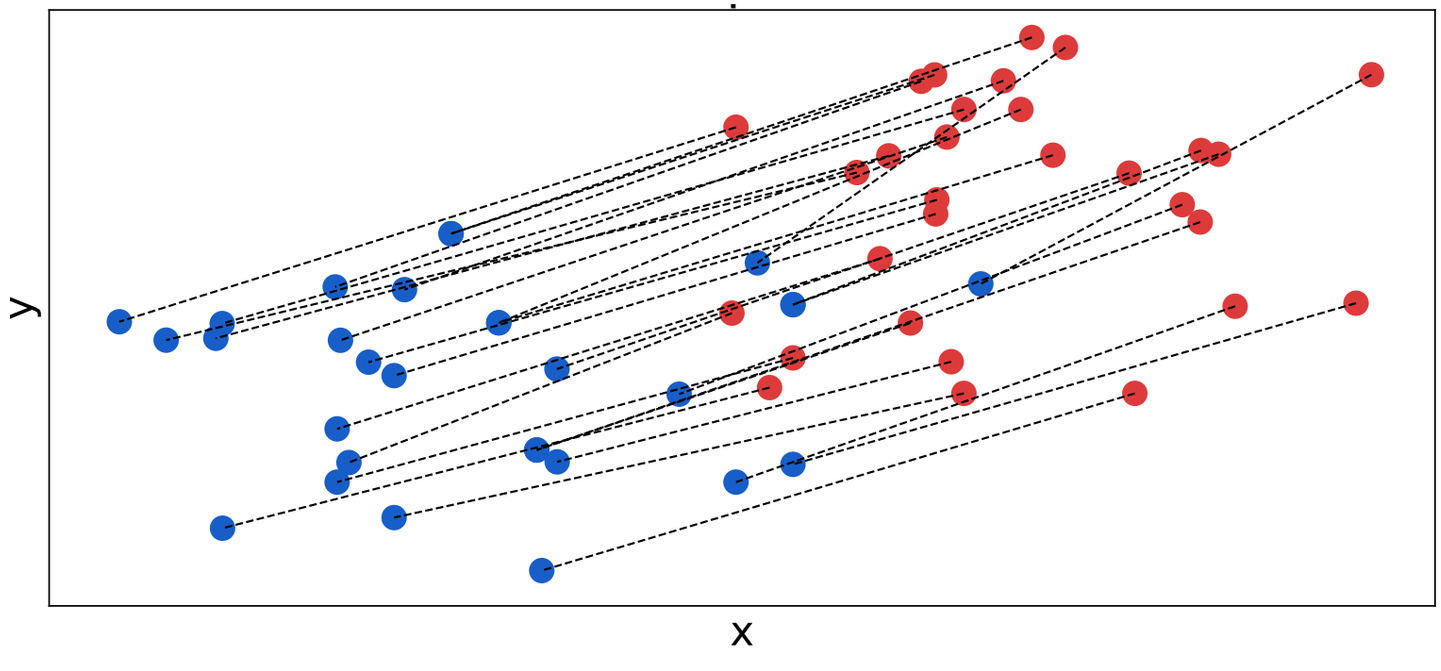
One-to-one

Fundamental theorem LP:

$$\boldsymbol{\pi}^* \leftrightarrow \sigma^* \in S_n$$

Optimal coupling is a permutation

Solves the Monge Problem



Solving OT

Entropic regularization

Discrete probability measures

$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i} \quad \nu = \sum_{j=1}^{m} b_j \delta_{y_j}$$

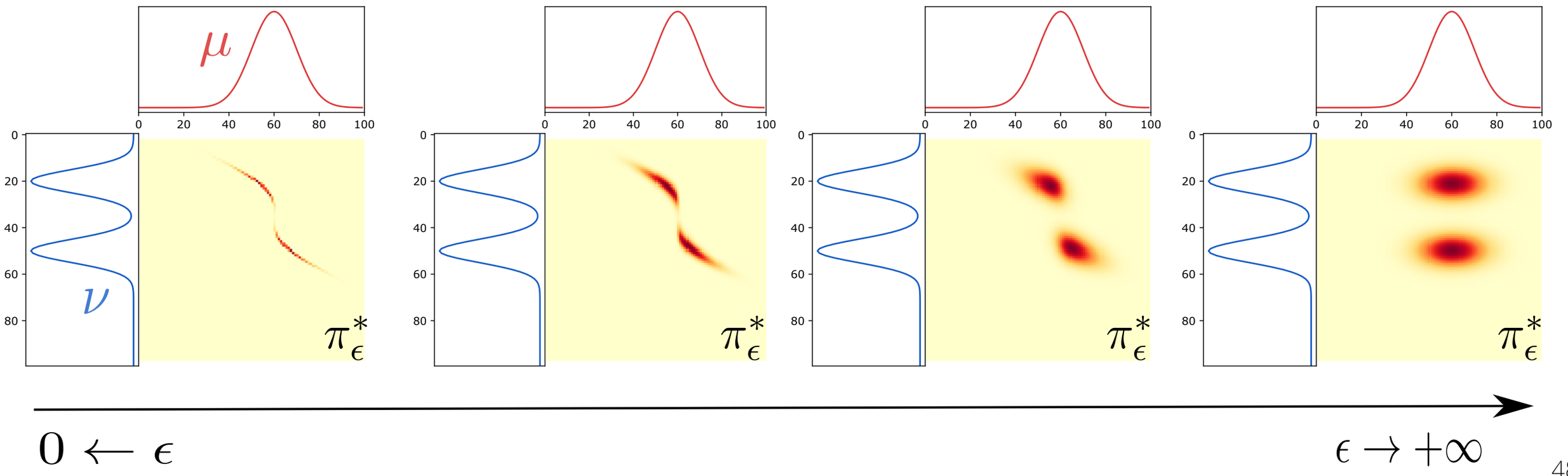
Strongly convex problem:

$$\min_{\boldsymbol{\pi} \in \Pi(\mathbf{a}, \mathbf{b})} \langle \mathbf{C}, \boldsymbol{\pi} \rangle - \varepsilon H(\boldsymbol{\pi})$$

Entropy term $H(\boldsymbol{\pi}) = -\sum_{ij} (\log(\pi_{ij}) - 1) \pi_{ij}$

Sinkhorn-Knopp algorithm: 1) fast 2) based on matrix multiplication

τ approximate solution $\sim O(n^2 \log(n) \tau^{-3})$



Solving OT

Entropic regularization

Discrete probability measures

$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i} \quad \nu = \sum_{j=1}^{m} b_j \delta_{y_j}$$

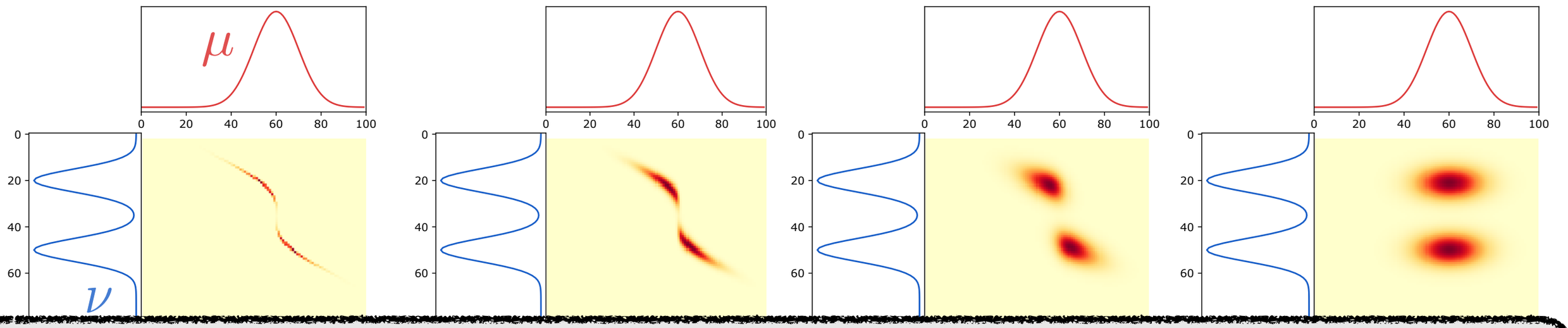
Strongly convex problem:

$$\min_{\boldsymbol{\pi} \in \Pi(\mathbf{a}, \mathbf{b})} \langle \mathbf{C}, \boldsymbol{\pi} \rangle - \varepsilon H(\boldsymbol{\pi})$$

Entropy term $H(\boldsymbol{\pi}) = -\sum_{ij} (\log(\pi_{ij}) - 1) \pi_{ij}$

Sinkhorn-Knopp algorithm: 1) fast 2) based on matrix multiplication

τ approximate solution $\sim O(n^2 \log(n) \tau^{-3})$



Linear OT: costly but solvable in practice

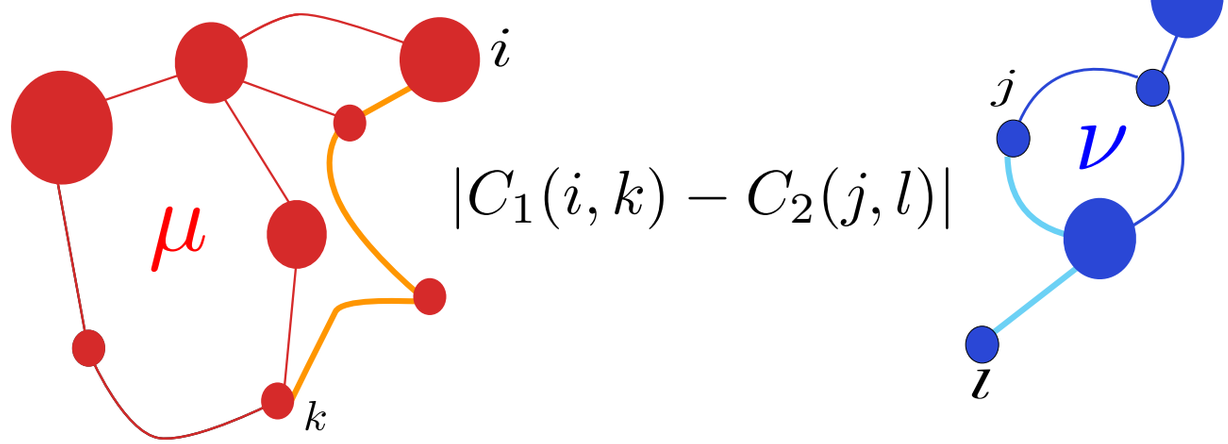
Solving OT

A quadratic problem (QP)

Discrete probability measures

$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i} \quad \nu = \sum_{j=1}^{m} b_j \delta_{y_j}$$

$$\mathcal{X}, \mathcal{Y} \not\subset \Omega$$



$$\min_{\boldsymbol{\pi} \in \Pi(\mathbf{a}, \mathbf{b})} \sum_{ijkl} |C_1(i,k) - C_2(j,l)|^p \pi_{ij} \pi_{kl}$$

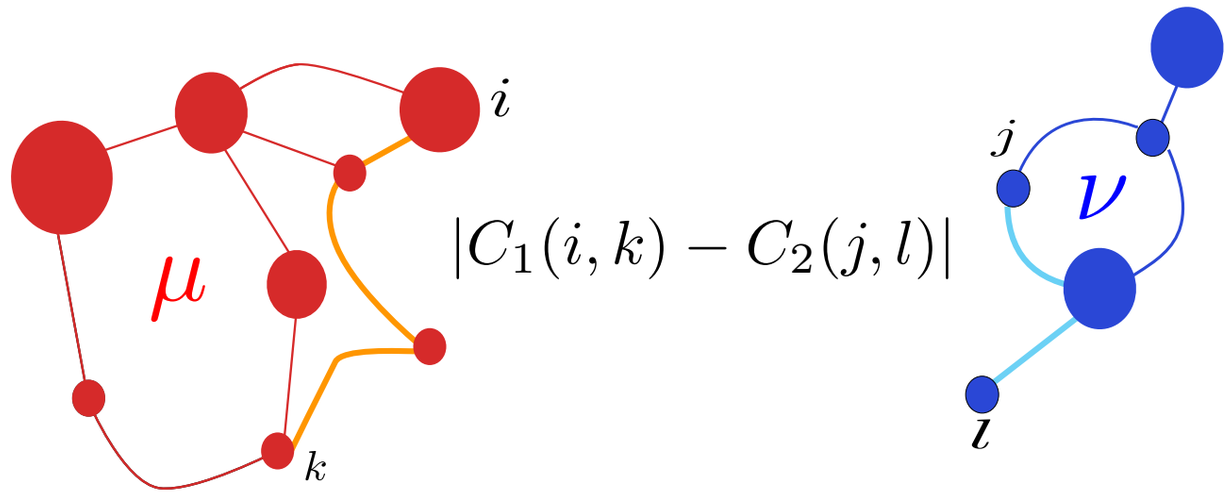
Solving OT

A quadratic problem (QP)

Discrete probability measures

$\mu = \sum_{i=1}^{n} a_i \delta_{x_i}$ $\nu = \sum_{j=1}^{m} b_j \delta_{y_j}$

$\mathcal{X}, \mathcal{Y} \not\subset \Omega$



$$\min_{\boldsymbol{\pi} \in \Pi(\mathbf{a},\mathbf{b})} \sum_{ijkl} |C_1(i,k) - C_2(j,l)|^p \pi_{ij} \pi_{kl}$$

Non convex QP: NP-hard in general (graph matching problem)

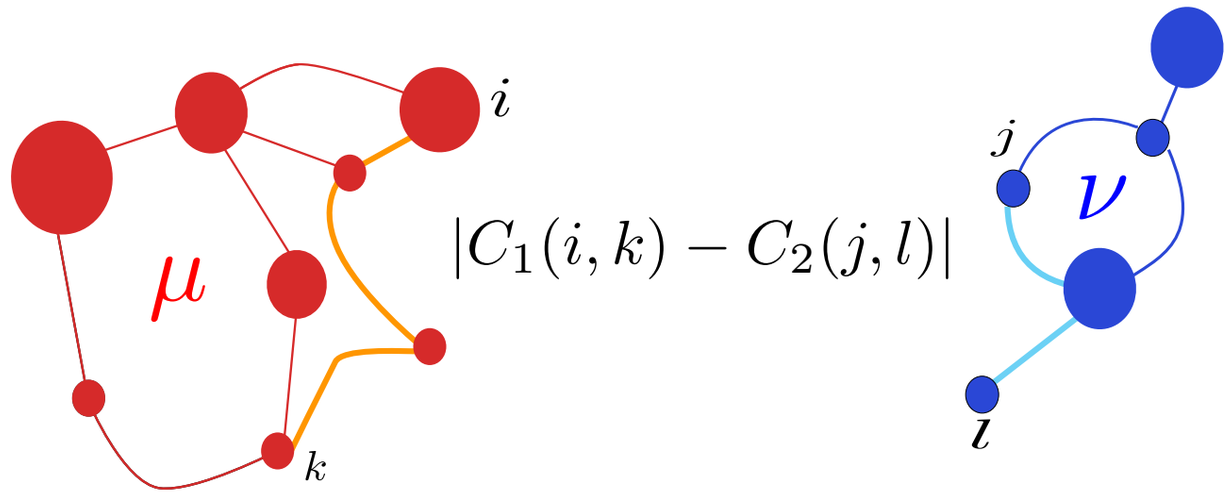
Solving OT

A quadratic problem (QP)

Discrete probability measures

$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i} \quad \nu = \sum_{j=1}^{m} b_j \delta_{y_j}$$

$$\mathcal{X}, \mathcal{Y} \not\subset \Omega$$



$$\min_{\boldsymbol{\pi} \in \Pi(\mathbf{a}, \mathbf{b})} \sum_{ijkl} |C_1(i,k) - C_2(j,l)|^p \pi_{ij} \pi_{kl} \; \boxed{-\varepsilon H(\boldsymbol{\pi})}$$

- Non convex QP: NP-hard in general
- With entropic regularization [Peyré 2016, Solomon 2016]
- Can be solved using projected gradient descent under KL geometry
- Each gradient step: Sinkhorn algorithm

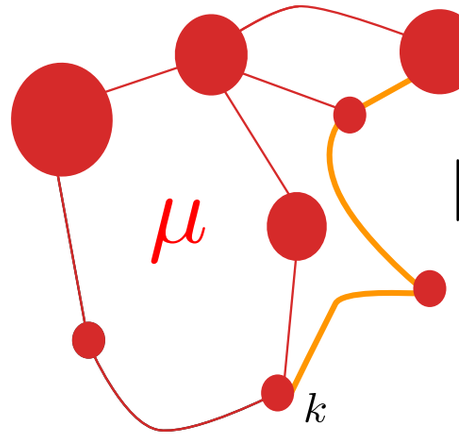
Solving OT

A quadratic problem (QP)

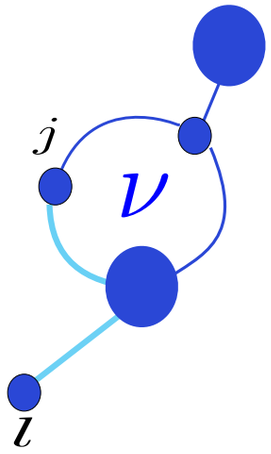
Discrete probability measures

$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i} \quad \nu = \sum_{j=1}^{m} b_j \delta_{y_j}$$

$$\mathcal{X}, \mathcal{Y} \not\subset \Omega$$



$$|C_1(i,k) - C_2(j,l)|$$



$$\min_{\boldsymbol{\pi} \in \Pi(\mathbf{a},\mathbf{b})} \sum_{ijkl} |C_1(i,k) - C_2(j,l)|^p \pi_{ij} \pi_{kl} \; -\varepsilon H(\boldsymbol{\pi})$$

- Non convex QP: NP-hard in general
- With entropic regularization [Peyré 2016, Solomon 2016] $\sim O(n_{iter} * n^2 \log(n))$
- Can be solved using projected gradient descent under KL geometry
- Each gradient step: Sinkhorn algorithm

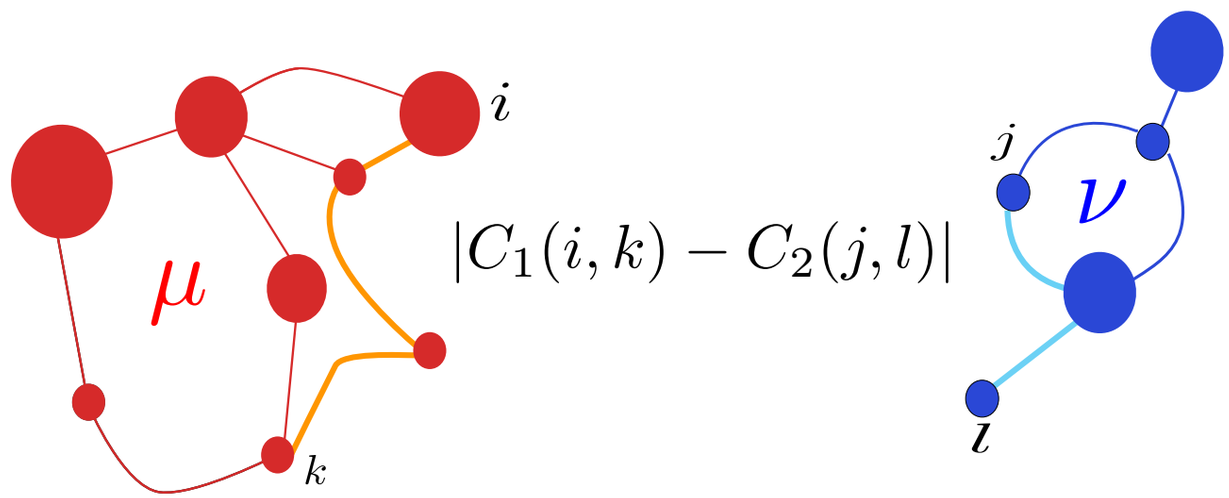
Solving OT

A quadratic problem (QP)

Discrete probability measures

$\mu = \sum_{i=1}^{n} a_i \delta_{x_i}$ $\nu = \sum_{j=1}^{m} b_j \delta_{y_j}$

$\mathcal{X}, \mathcal{Y} \not\subset \Omega$



$$\min_{\boldsymbol{\pi} \in \Pi(\mathbf{a},\mathbf{b})} \sum_{ijkl} |C_1(i,k) - C_2(j,l)|^p \pi_{ij} \pi_{kl}$$

- Non convex QP: NP-hard in general
- With entropic regularization [Peyré 2016, Solomon 2016] $\sim O(n_{iter} * n^2 \log(n))$
- Can be solved using projected gradient descent under KL geometry
- Each gradient step: Sinkhorn algorithm

Hard to solve and even to approximate…

Solving OT

Computing GW

Solving FGW: a non convex QP

$$\min_{\boldsymbol{\pi} \in \Pi(\mathbf{a},\mathbf{b})} \sum_{ijkl} |C_1(i,k) - C_2(j,l)|^p \pi_{ij} \pi_{kl}$$

Quadratic function over polytope -> Conditional Gradient algorithm (a.k.a Frank-Wolfe)

Non convex but converges to a **local optimal solution** [Lacoste-Julien 2016]

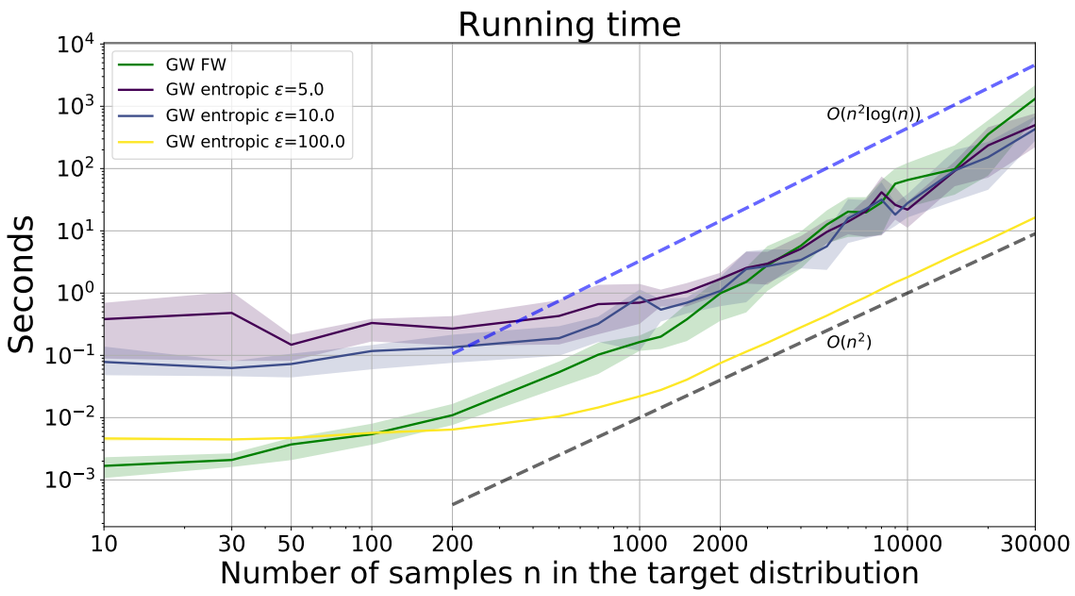
Find a **sparse** solution. FW gap = $O(\frac{1}{\sqrt{n_{iter}}})$

Algorithm 1 Conditional Gradient (CG) for FGW

$\pi^{(0)} \leftarrow \mathbf{h}\mathbf{g}^\top$
for $i = 1, \ldots,$ **do**
  $\mathbf{G} \leftarrow$ Gradient from GW loss $w.r.t.$ $\boldsymbol{\pi}^{(i-1)}$
  $\tilde{\boldsymbol{\pi}}^{(i)} \leftarrow$ Solve OT with ground loss $\mathbf{G}$
  $\tau^{(i)} \leftarrow$ Line-search for GW loss with $\tau \in (0, 1)$ (closed-form)
  $\boldsymbol{\pi}^{(i)} \leftarrow (1 - \tau^{(i)})\boldsymbol{\pi}^{(i-1)} + \tau^{(i)}\tilde{\boldsymbol{\pi}}^{(i)}$
end for

Complexity

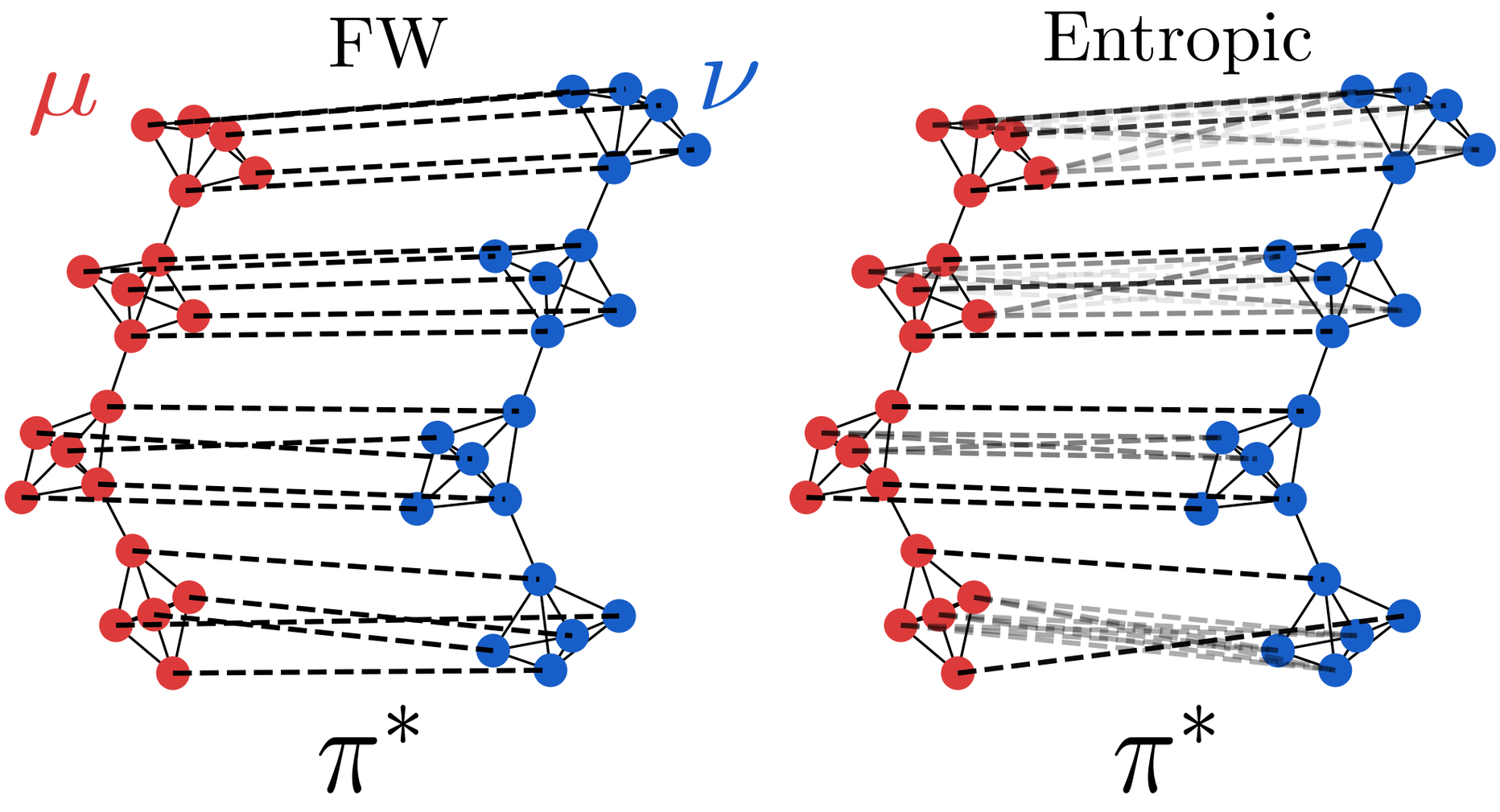
$$O(n_{iter}\ n^3)$$



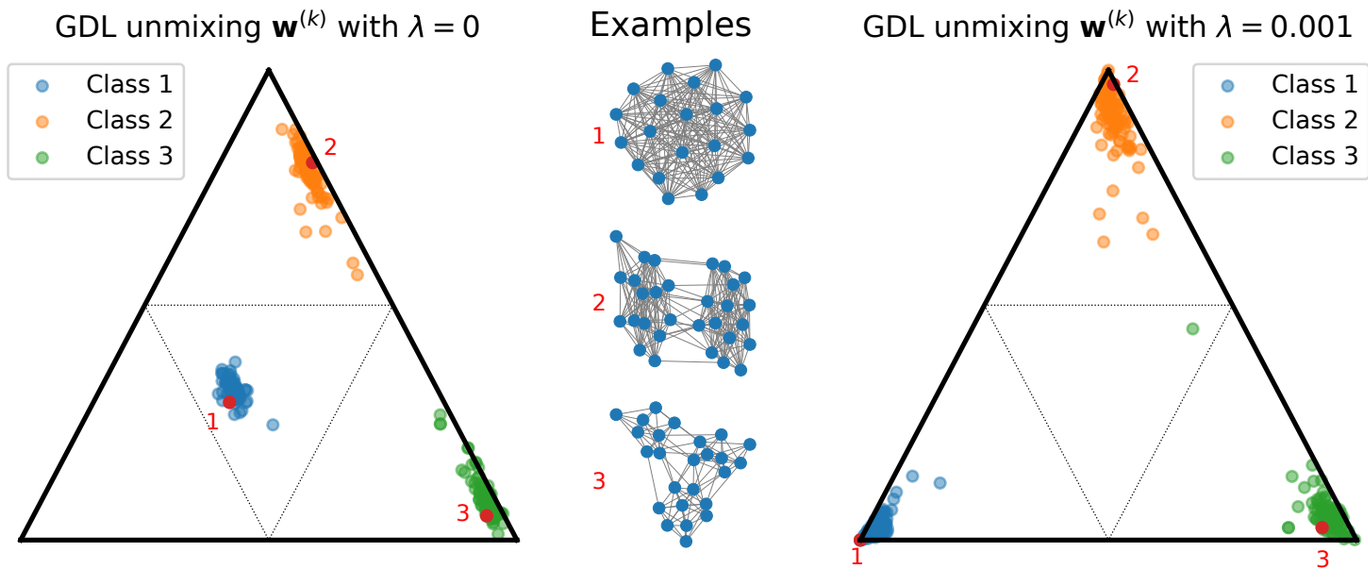
...to Gromov-Wasserstein

An example on graphs

$\mathbf{C}_1, \mathbf{C}_2$ are the shortest path distance in each graph



Online graph Dictionary Learning



From Euclidean spaces…

Traditional dictionary learning

One point: $\mathbf{x} \in \mathbb{R}^m$

$$\mathbf{x} = w_1\mathbf{d}_1 + \cdots + w_d\mathbf{d}_d = \mathbf{D}\mathbf{w}$$

Dictionary

$\mathbf{D} \in \mathbb{R}^{m \times d}$

Overcomplete:

$m << d$

Unmixing

$\mathbf{w} \in \mathbb{R}^d$

From Euclidean spaces…

Traditional dictionary learning

One point: $\mathbf{x} \in \mathbb{R}^m$

$$\mathbf{x} = w_1\mathbf{d}_1 + \cdots + w_d\mathbf{d}_d = \mathbf{D}\mathbf{w}$$

Atom

Dictionary

$\mathbf{D} \in \mathbb{R}^{m \times d}$

Overcomplete:

$m << d$

Unmixing

$\mathbf{w} \in \mathbb{R}^d$

From Euclidean spaces…

Traditional dictionary learning

One point: $\mathbf{x} \in \mathbb{R}^m$

$$\mathbf{x} = w_1 \mathbf{d}_1 + \cdots + w_d \mathbf{d}_d = \mathbf{D}\mathbf{w}$$

Dictionary

$\mathbf{D} \in \mathbb{R}^{m \times d}$

Overcomplete:

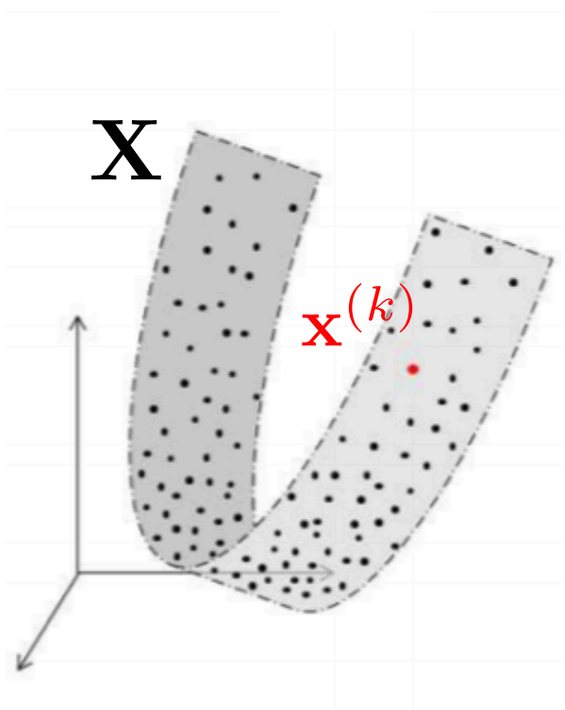
$m << d$

Unmixing

$\mathbf{w} \in \mathbb{R}^d$

Many points = dataset: $\mathbf{X} \in \mathbb{R}^{m \times K}$

$$\forall k, \mathbf{x}^{(k)} = w_1^{(k)} \mathbf{d}_1 + \cdots + w_d^{(k)} \mathbf{d}_d$$



From Euclidean spaces…

Traditional dictionary learning

Comes with many flavors…

One point: $\mathbf{x} \in \mathbb{R}^m$

$$\mathbf{x} = w_1\mathbf{d}_1 + \cdots + w_d\mathbf{d}_d = \mathbf{D}\mathbf{w}$$

Dictionary

$\mathbf{D} \in \mathbb{R}^{m \times d}$

- $\mathbf{D} \geq 0$
- $\|\mathbf{D}\|_F \leq 1$

Unmixing

$\mathbf{w} \in \mathcal{U}$

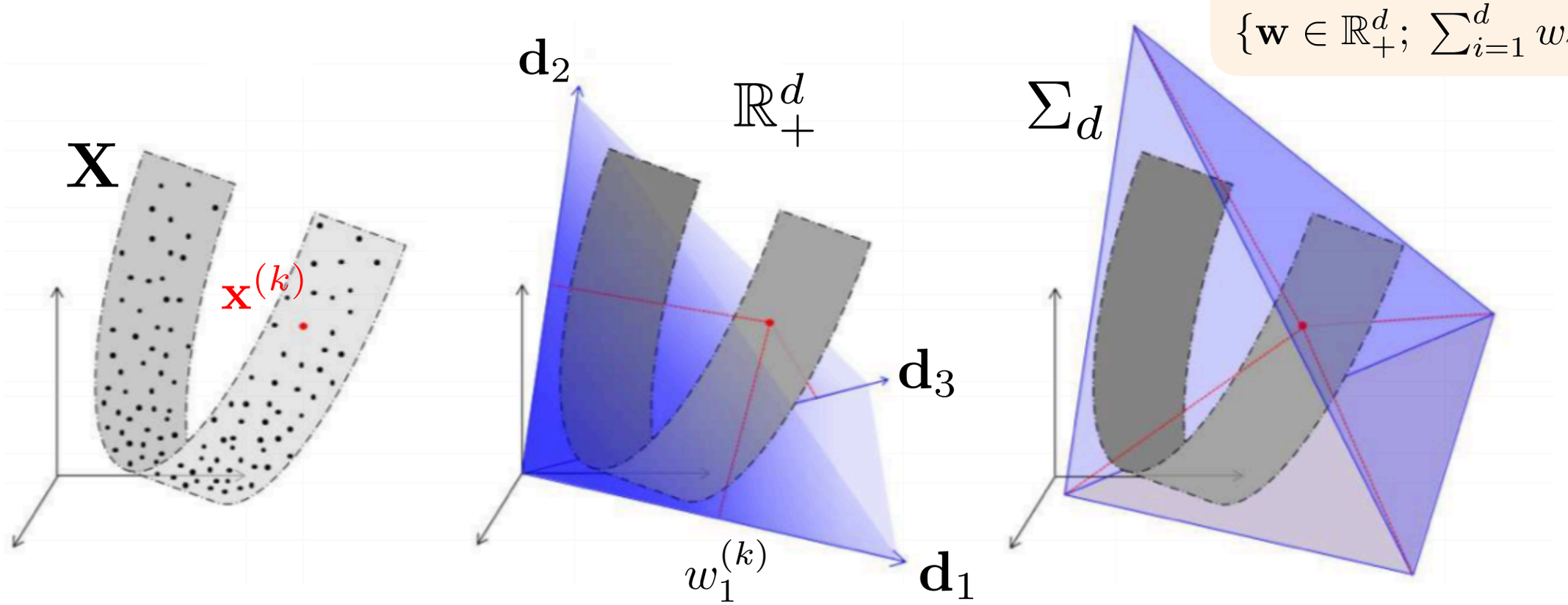
$\mathcal{U} \in \{\mathbb{R}^d, \mathbb{R}^d_+, \Sigma_d\}$

Many points = dataset: $\mathbf{X} \in \mathbb{R}^{m \times K}$

$$\forall k, \mathbf{x}^{(k)} = w_1^{(k)}\mathbf{d}_1 + \cdots + w_d^{(k)}\mathbf{d}_d$$

Simplex:

$\{\mathbf{w} \in \mathbb{R}^d_+;\ \sum_{i=1}^d w_i = 1\}$



From Euclidean spaces…

Traditional dictionary learning

Comes with many flavors…

One point: $\mathbf{x} \in \mathbb{R}^m$

$$\mathbf{x} = w_1\mathbf{d}_1 + \cdots + w_d\mathbf{d}_d = \mathbf{D}\mathbf{w}$$

Dictionary

$$\mathbf{D} \in \mathbb{R}^{m \times d}$$

- $\mathbf{D} \geq 0$
- $\|\mathbf{D}\|_F \leq 1$

Unmixing

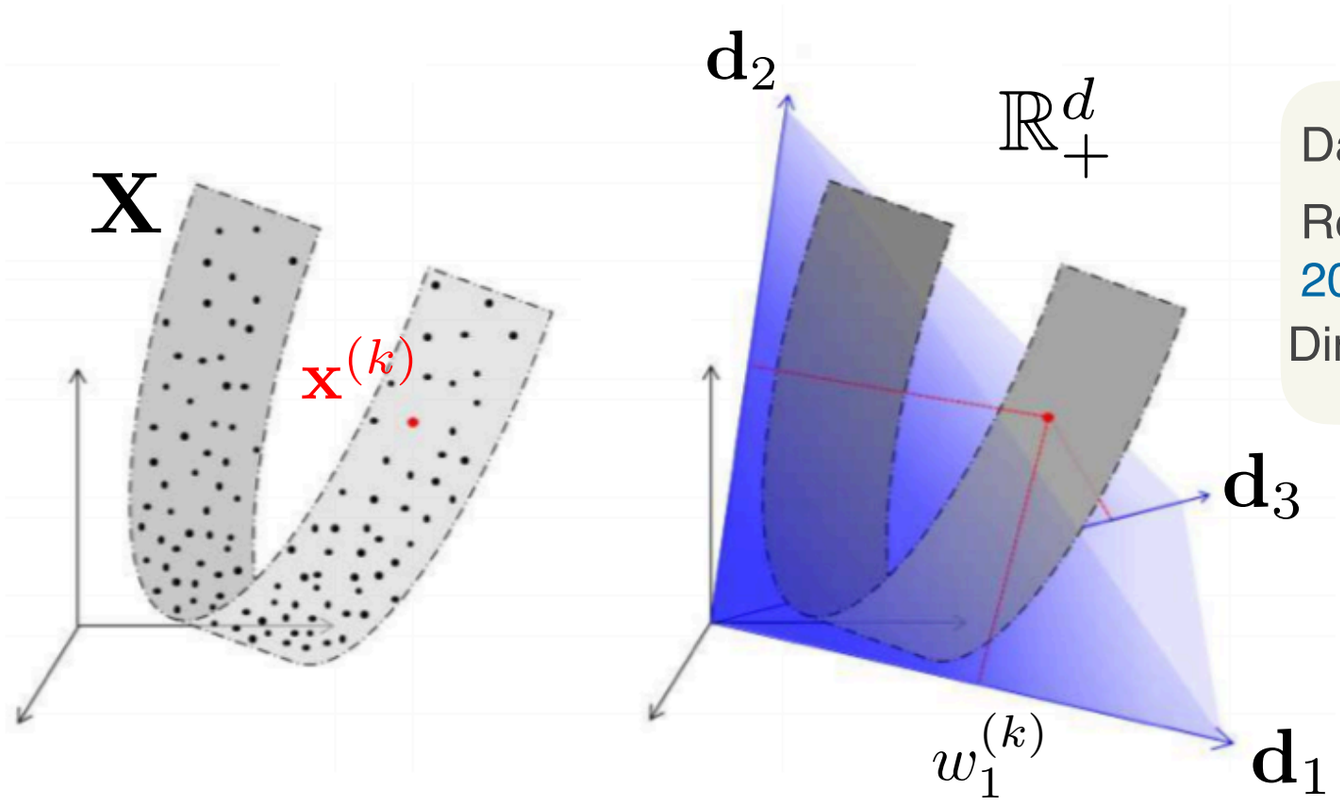
$$\mathbf{w} \in \mathcal{U}$$

$$\mathcal{U} \in \{\mathbb{R}^d, \mathbb{R}^d_+, \Sigma_d\}$$

Many points = dataset: $\mathbf{X} \in \mathbb{R}^{m \times K}$

$$\forall k, \mathbf{x}^{(k)} = w_1^{(k)}\mathbf{d}_1 + \cdots + w_d^{(k)}\mathbf{d}_d$$

Representation learning



Data clustering [Ng et al., 2002]

Recommendation systems [Bobadilla et al., 2013]

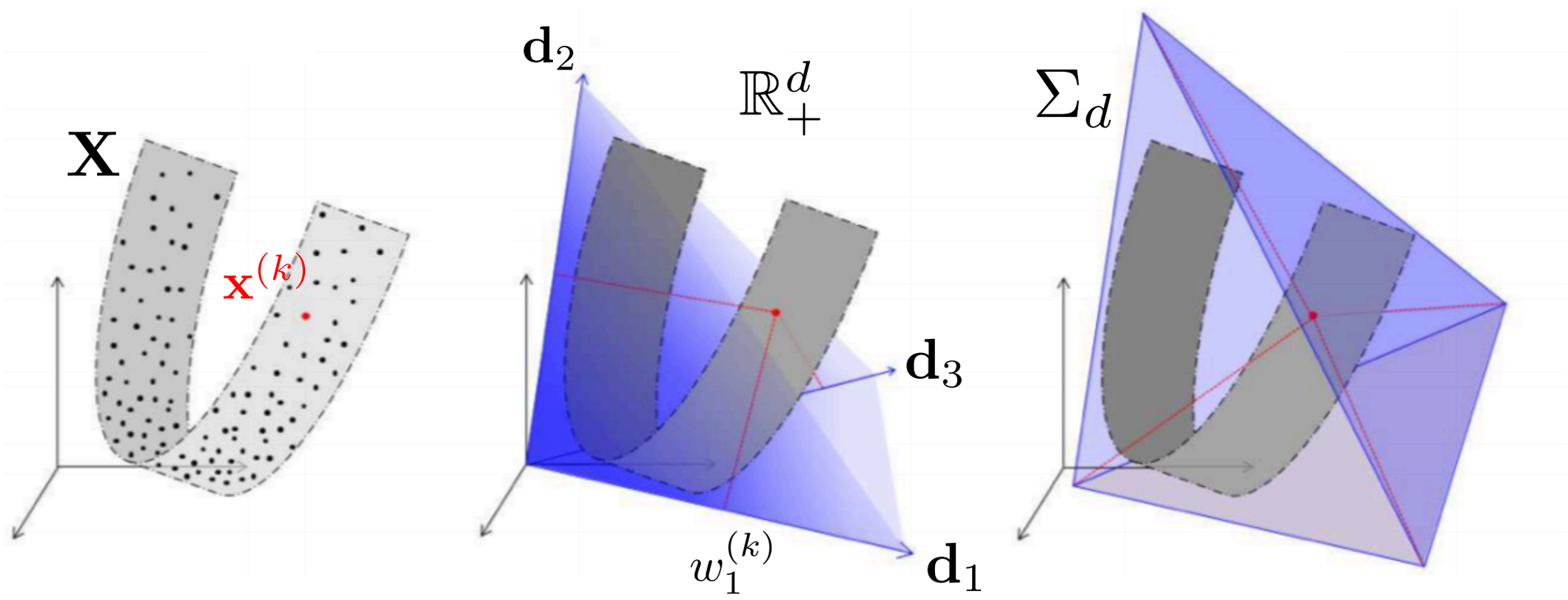
Dimensionality reduction [Candes` et al., 2011]

From Euclidean spaces…

Traditional dictionary learning

Solve:

$$\min_{\substack{\forall k \in [\![K]\!],\ \mathbf{w}^{(k)} \in \mathcal{U} \\ \mathbf{D} \in \mathcal{D}}} \ \frac{1}{K} \sum_{k=1}^{K} \|\mathbf{x}^{(k)} - \sum_{i=1}^{d} w_i^{(k)} \mathbf{d}_i\|_2^2 + R(\mathbf{w}^{(k)})$$



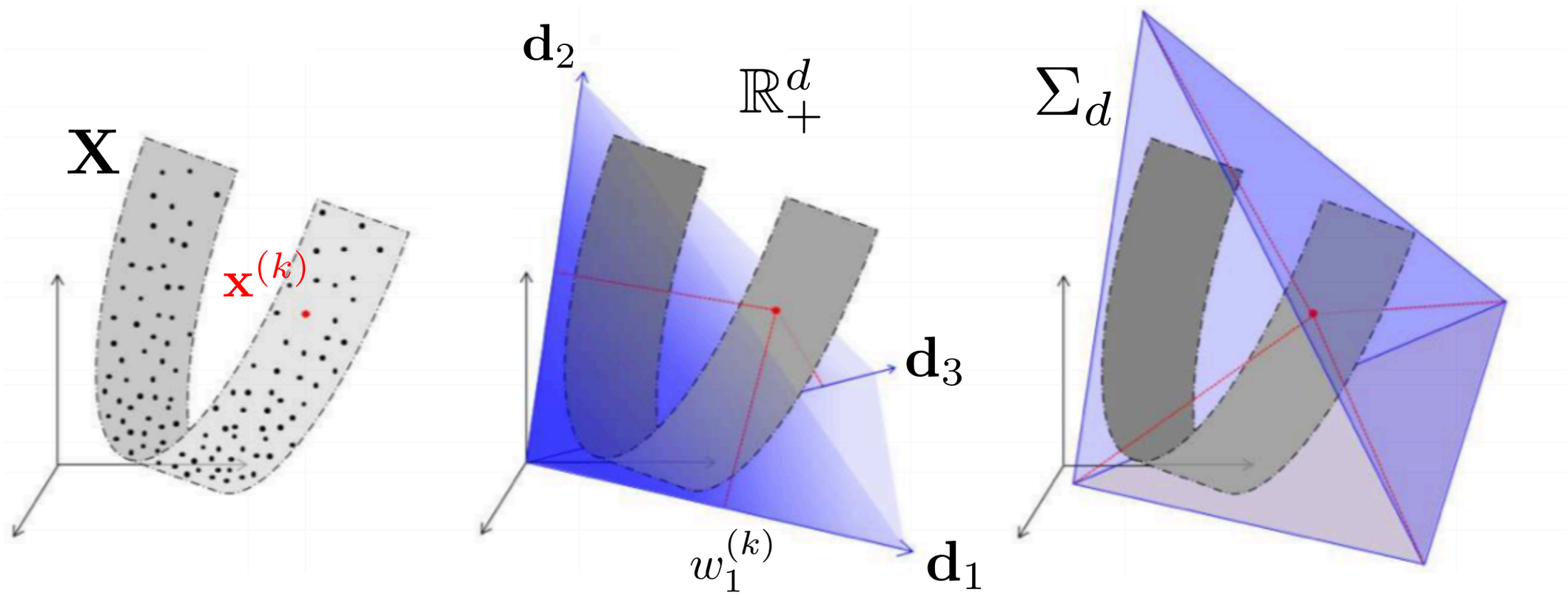
From Euclidean spaces…

Traditional dictionary learning

Solve:

$$\min_{\substack{\forall k \in [\![K]\!],\ \mathbf{w}^{(k)} \in \mathcal{U} \\ \mathbf{D} \in \mathcal{D}}} \frac{1}{K} \sum_{k=1}^{K} \|\mathbf{x}^{(k)} - \sum_{i=1}^{d} w_i^{(k)} \mathbf{d}_i\|_2^2 + R(\mathbf{w}^{(k)})$$

Representation of $\mathbf{x}^{(k)}$



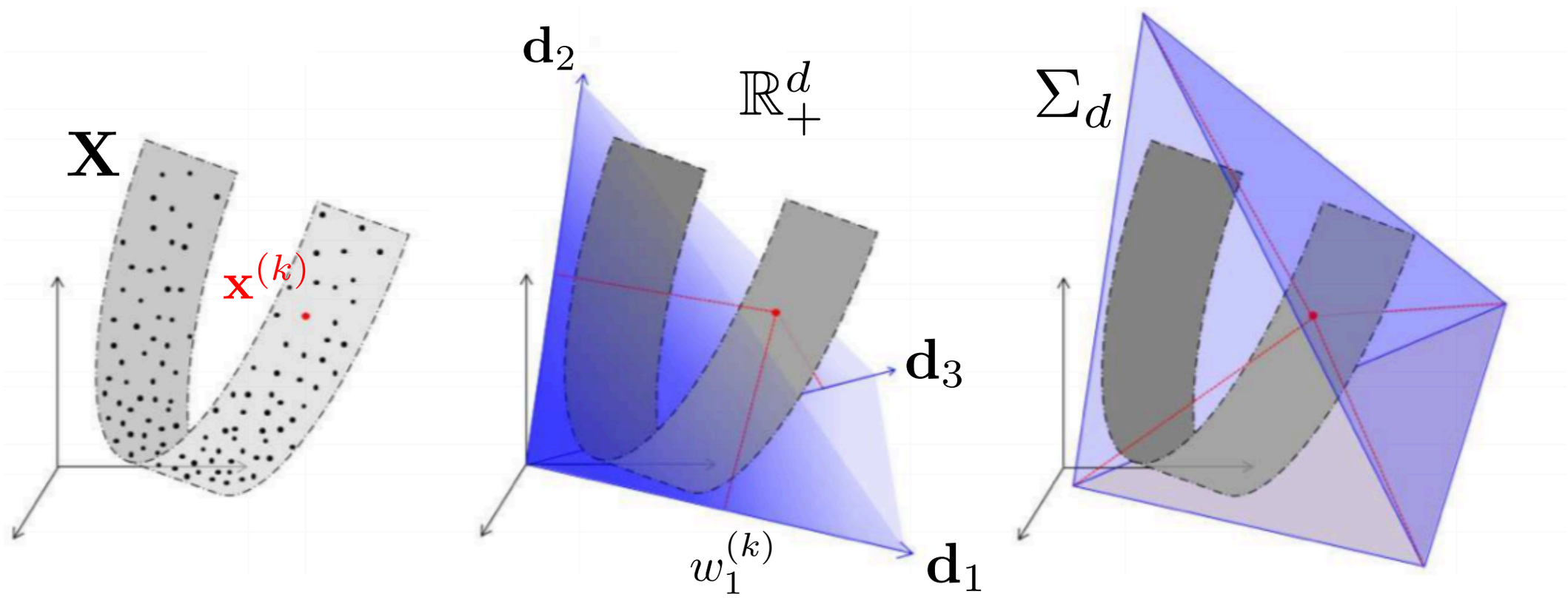
From Euclidean spaces…

Traditional dictionary learning

Solve:

$$\min_{\substack{\forall k \in [\![K]\!],\ \mathbf{w}^{(k)} \in \mathcal{U} \\ \mathbf{D} \in \mathcal{D}}} \frac{1}{K} \sum_{k=1}^{K} \|\mathbf{x}^{(k)} - \sum_{i=1}^{d} w_i^{(k)} \mathbf{d}_i\|_2^2 + R(\mathbf{w}^{(k)})$$

Fidelity term wrt to the dataset



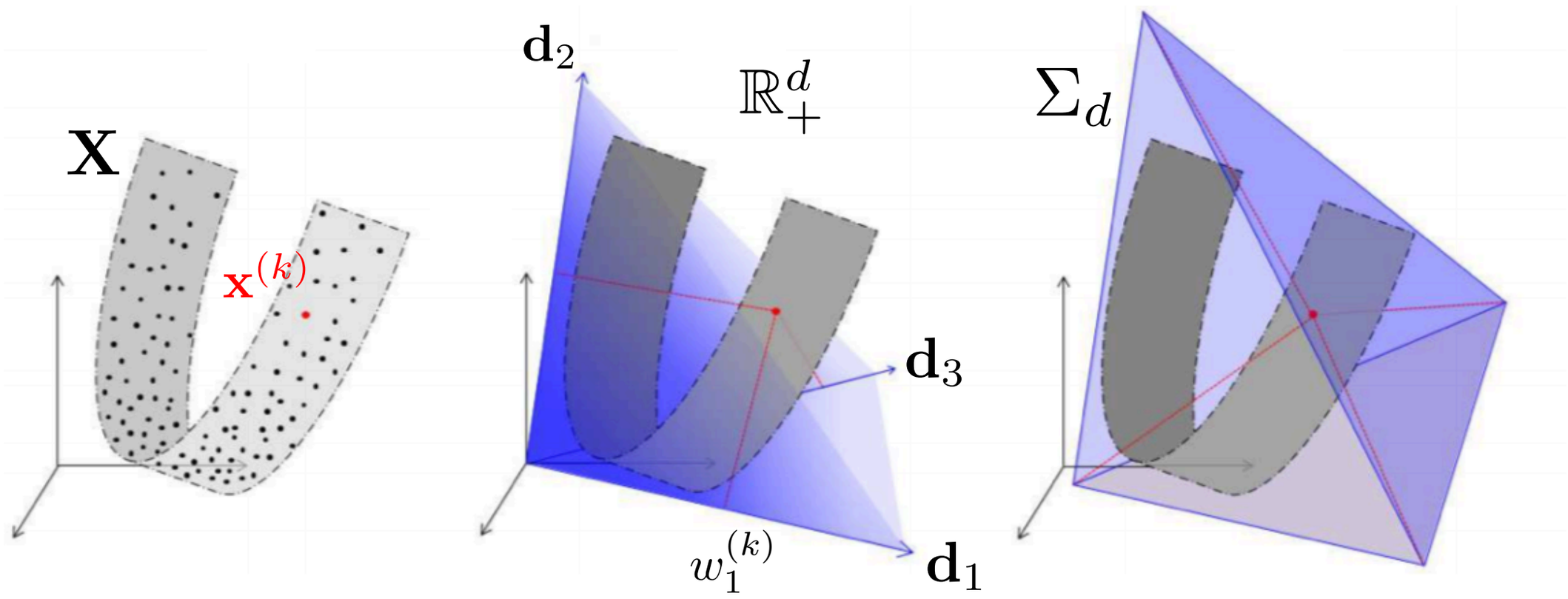
From Euclidean spaces…

Traditional dictionary learning

Solve:

$$\min_{\substack{\forall k \in [\![K]\!],\ \mathbf{w}^{(k)} \in \mathcal{U} \\ \mathbf{D} \in \mathcal{D}}} \frac{1}{K} \sum_{k=1}^{K} \|\mathbf{x}^{(k)} - \sum_{i=1}^{d} w_i^{(k)} \mathbf{d}_i\|_2^2 + R(\mathbf{w}^{(k)})$$

Find the dico and the representations (unmixings) that best fit our data



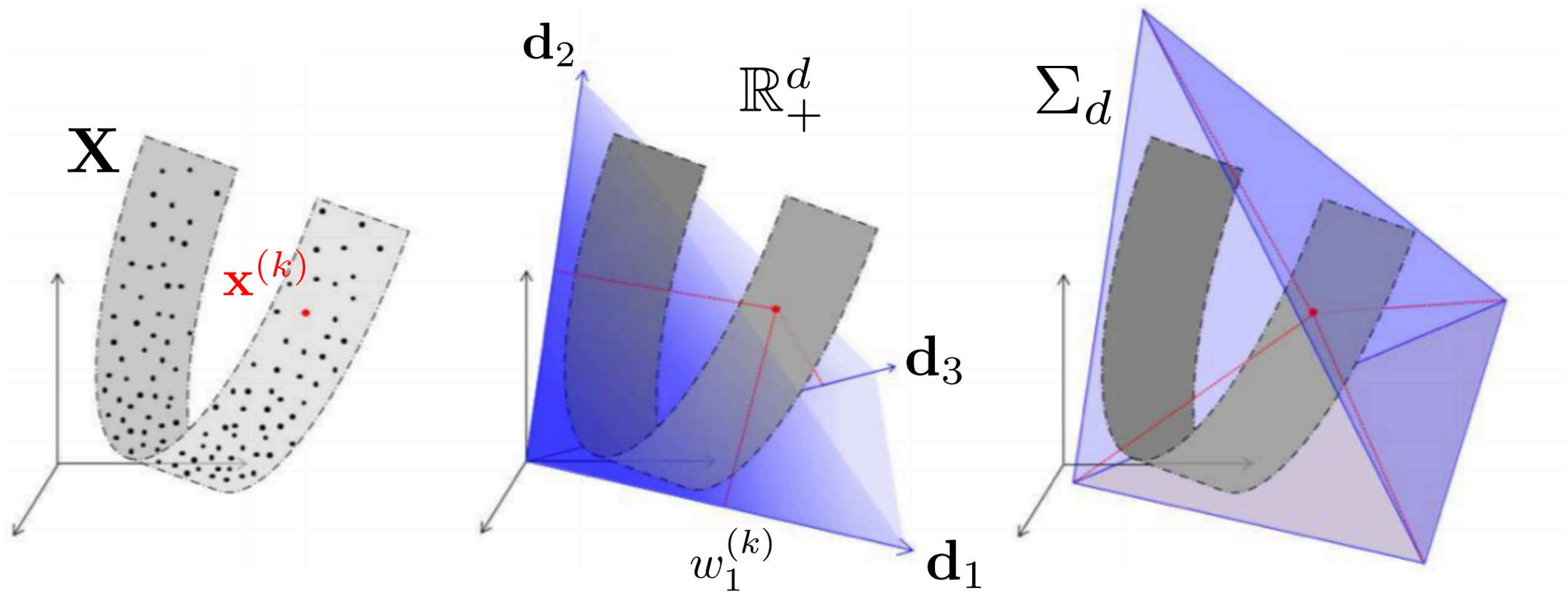
From Euclidean spaces…

Traditional dictionary learning

Solve:

$$\min_{\substack{\forall k \in [\![K]\!],\ \mathbf{w}^{(k)} \in \mathcal{U} \\ \mathbf{D} \in \mathcal{D}}} \frac{1}{K} \sum_{k=1}^{K} \|\mathbf{x}^{(k)} - \sum_{i=1}^{d} w_i^{(k)} \mathbf{d}_i\|_2^2 + R(\mathbf{w}^{(k)})$$

Regularization e.g sparsity (mandatory for the overcomplete case !)



From Euclidean spaces…

Traditional dictionary learning

Solve:

$$\min_{\substack{\forall k \in [\![K]\!],\ \mathbf{w}^{(k)} \in \mathcal{U} \\ \mathbf{D} \in \mathcal{D}}} \ \frac{1}{K} \sum_{k=1}^{K} \|\mathbf{x}^{(k)} - \sum_{i=1}^{d} w_i^{(k)} \mathbf{d}_i\|_2^2 + R(\mathbf{w}^{(k)})$$

Alternate Optimization:

Update $\mathbf{D}$: $\min_{\mathbf{D} \in \mathcal{D}} \ \|\mathbf{x}^{(k)} - \sum_{i=1}^{d} w_i^{(k)} \mathbf{d}_i\|_2^2$

Method of Optimal Directions [Engan et al., 1999]

K-SVD [Aharon et al. 2006]

Online L1: stoch. gradient [Mairal et al., 2010]

NMF [Lee & Seung 1999]

Update $\mathbf{W}$: $\min_{\forall k \in [\![K]\!],\ \mathbf{w}^{(k)} \in \mathcal{U}} \ \frac{1}{K} \sum_{k=1}^{K} \|\mathbf{x}^{(k)} - \sum_{i=1}^{d} w_i^{(k)} \mathbf{d}_i\|_2^2 + R(\mathbf{w}^{(k)})$

Strategies: K big

- Per batch of data: sampling
- Online: one sample selected

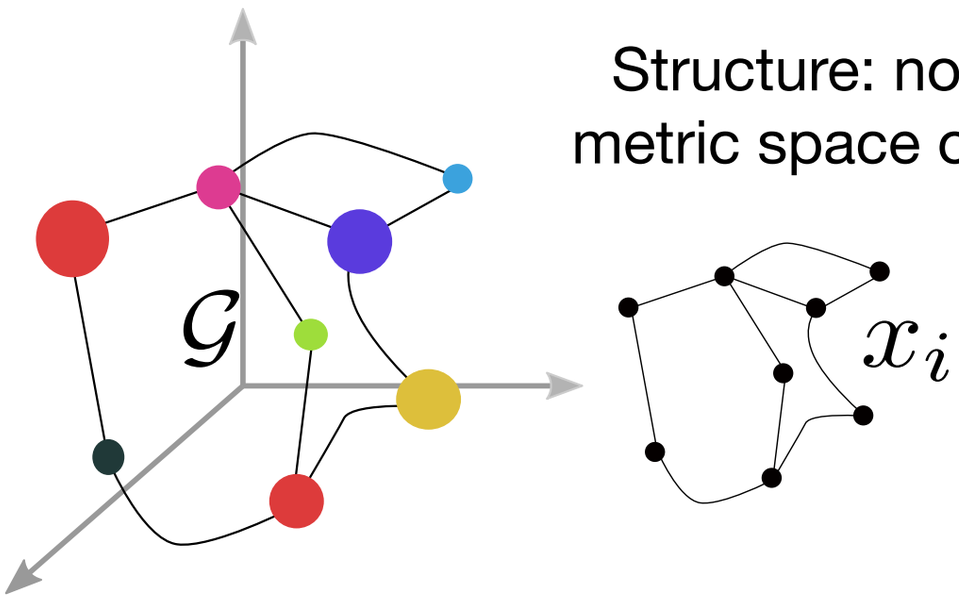
l1 minimization, orthogonal matching pursuit …

To structured data…

Gromov-Wasserstein dictionary learning

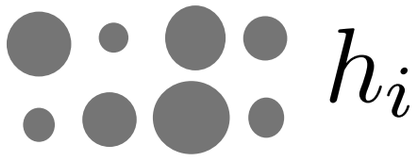
Setting:

Features  $a_i \in \Omega$



Structure: nodes in the metric space of the graph

Weights



To structured data…

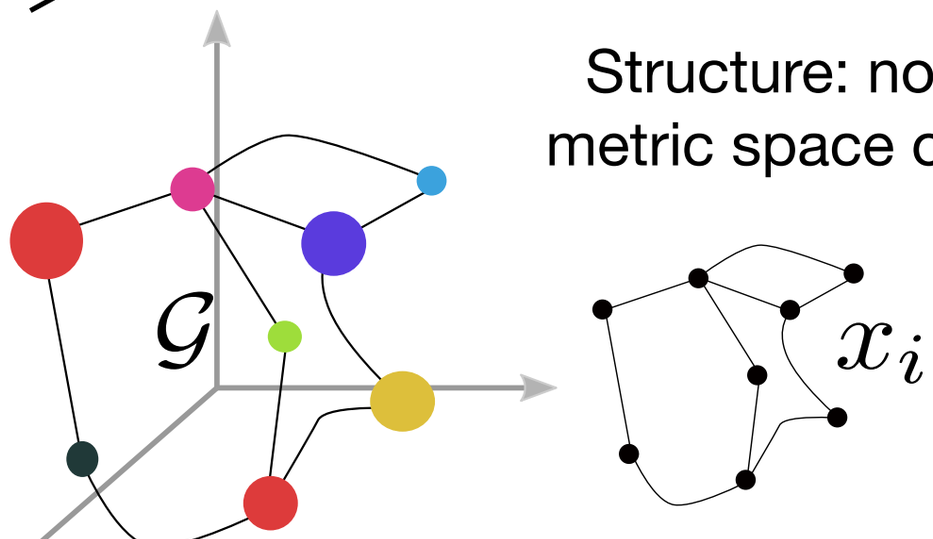
Gromov-Wasserstein dictionary learning

Setting:

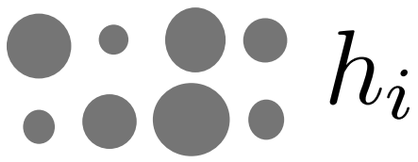
for simplicity…

Features $a_i \in \Omega$

Structure: nodes in the metric space of the graph



Weights



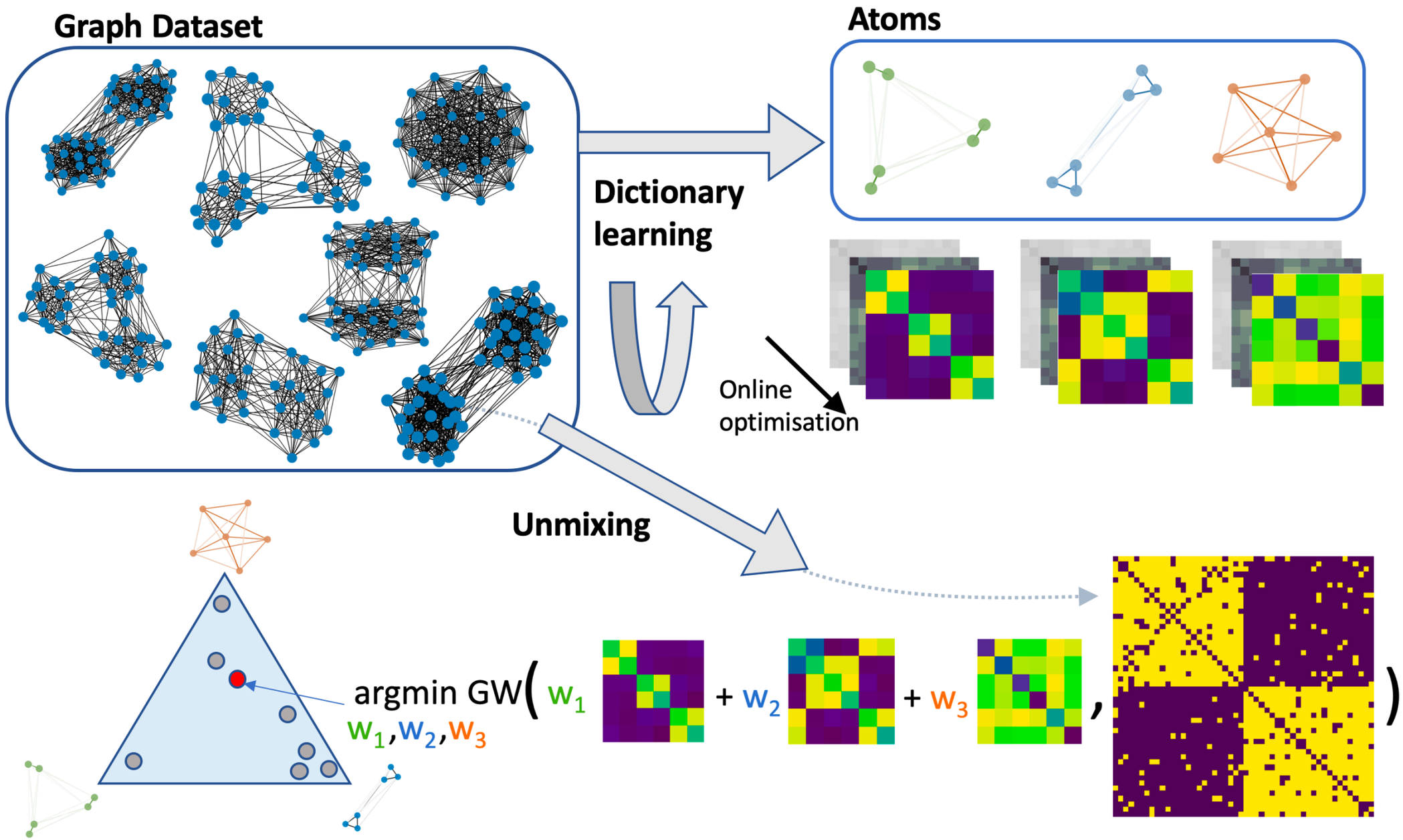
$$\mathcal{G} = (\mathbf{C}, \mathbf{h})$$

$\mathbf{C}$ pairwise relation matrix (adjacency, shortest-path, laplacian …)

To structured data…

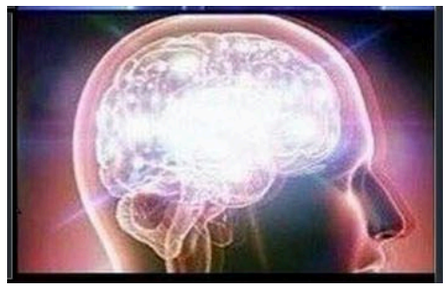
Gromov-Wasserstein dictionary learning

Big picture:



To structured data…

Gromov-Wasserstein dictionary learning



Step 1. Linear modeling of a graph $\mathcal{G} = (\mathbf{C}, \mathbf{h})$

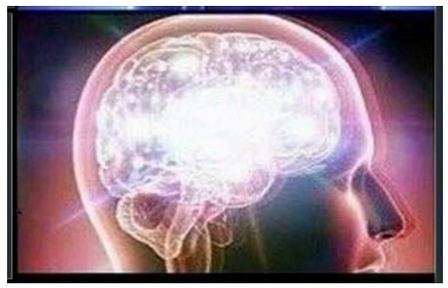
$$\min_{\boldsymbol{w} \in \Sigma_d} GW_2^2 \left(\boldsymbol{C}, \sum_{i=1}^{d} w_i \overline{\boldsymbol{C}}_i \right) - \lambda \|\boldsymbol{w}\|_2^2$$

Dictionary: $(\overline{\boldsymbol{C}}_1, \ldots, \overline{\boldsymbol{C}}_d)$

Atom: $\overline{\boldsymbol{C}}_i \in \mathbb{R}^{N \times N}$

To structured data…

Gromov-Wasserstein dictionary learning



Step 1. Linear modeling of a graph $\mathcal{G} = (\mathbf{C}, \mathbf{h})$

$$\min_{\boldsymbol{w} \in \Sigma_d} GW_2^2\left(\boldsymbol{C}, \sum_{i=1}^{d} w_i \overline{\boldsymbol{C}}_i\right) + \lambda \|\boldsymbol{w}\|_2^2$$

Dictionary: $(\overline{\boldsymbol{C}}_1, \ldots, \overline{\boldsymbol{C}}_d)$

Atom: $\overline{\boldsymbol{C}}_i \in \mathbb{R}^{N \times N}$

Fidelity term

↓ **produce**

$$\mathbf{C} \approx \sum_{i=1}^{d} w_i \overline{\mathbf{C}}_i$$

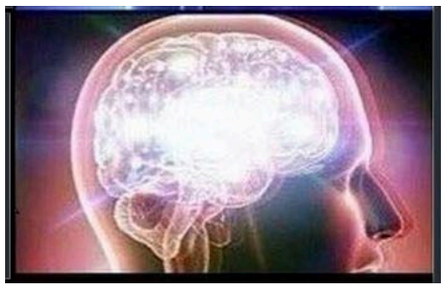
« like » in euclidean setting

$$\mathbf{x} = w_1 \mathbf{d}_1 + \cdots + w_d \mathbf{d}_d$$

!! Here not same sized matrices !!

To structured data…

Gromov-Wasserstein dictionary learning



Step 1. Linear modeling of a graph $\mathcal{G} = (\mathbf{C}, \mathbf{h})$

$$\min_{\boldsymbol{w} \in \Sigma_d} GW_2^2\left(\boldsymbol{C}, \sum_{i=1}^{d} w_i \overline{\boldsymbol{C}}_i\right) + \lambda \|\boldsymbol{w}\|_2^2$$

Dictionary: $(\overline{\boldsymbol{C}}_1, \ldots, \overline{\boldsymbol{C}}_d)$

Atom: $\overline{\boldsymbol{C}}_i \in \mathbb{R}^{N \times N}$

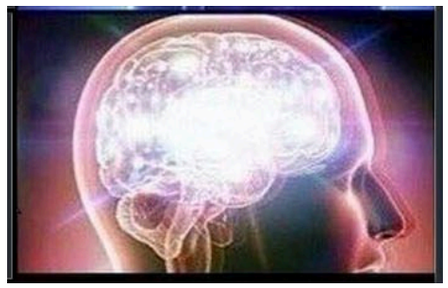
Interpretation when atoms are adjacency matrices

$$\sum_{i=1}^{d} w_i \overline{\mathbf{C}_i} \longrightarrow \text{probability of connections between the nodes}$$

$$w_i \in [0, 1]$$

To structured data…

Gromov-Wasserstein dictionary learning



Step 1. Linear modeling of a graph $\mathcal{G} = (\mathbf{C}, \mathbf{h})$

$$\min_{\boldsymbol{w} \in \Sigma_d} GW_2^2\left(\boldsymbol{C}, \sum_{i=1}^{d} w_i \overline{\boldsymbol{C}}_i\right) - \lambda \|\boldsymbol{w}\|_2^2$$

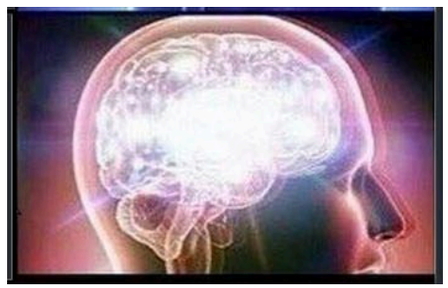
Dictionary: $(\overline{\boldsymbol{C}}_1, \ldots, \overline{\boldsymbol{C}}_d)$

Atom: $\overline{\boldsymbol{C}}_i \in \mathbb{R}^{N \times N}$

Promote sparsity in the simplex [Li and al. 2016]

To structured data…

Gromov-Wasserstein dictionary learning



Step 1. Linear modeling of a graph $\mathcal{G} = (\mathbf{C}, \mathbf{h})$

$$\min_{\boldsymbol{w} \in \Sigma_d} GW_2^2 \left(\boldsymbol{C}, \sum_{i=1}^{d} w_i \overline{\boldsymbol{C}}_i \right) - \lambda \|\boldsymbol{w}\|_2^2$$

Algorithm for **unmixing**:

1: Initialize $\boldsymbol{w} = \frac{1}{d}\mathbf{1}_d$
2: **repeat**
3: Compute OT matrix $\boldsymbol{T}$ of $GW_2^2 \left(\boldsymbol{C}, \sum_s w_s \overline{\boldsymbol{C}}_s \right)$, with CG algorithm
4: Compute the optimal $\boldsymbol{w}$ for a fixed $\boldsymbol{T}$ with CG algorithm.
5: **until convergence**

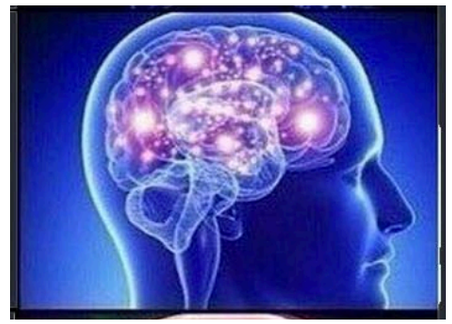
Few iter in practice

CG/Frank Wolfe on the simplex

$O(d)$

To structured data…

Gromov-Wasserstein dictionary learning



Step 1. Linear modeling of a graph $\mathcal{G} = (\mathbf{C}, \mathbf{h})$

$$\min_{\boldsymbol{w} \in \Sigma_d} GW_2^2 \left(\boldsymbol{C}, \sum_{i=1}^d w_i \overline{\boldsymbol{C}}_i \right) - \lambda \|\boldsymbol{w}\|_2^2$$

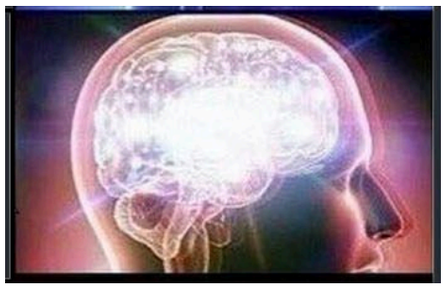
Step 2. Linear modeling of a bunch of graphs $\{\mathcal{G}^k : (\mathbf{C}^{(k)}, \mathbf{h}^{(k)})\}_{k \in [\![K]\!]}$

$$\min_{\left\{\boldsymbol{w}^{(k)}\right\}_{k \in [K]}} \sum_{k=1}^K GW_2^2 \left(\boldsymbol{C}^{(k)}, \sum_{i=1}^d w_i^{(k)} \overline{\boldsymbol{C}}_i \right) - \lambda \left\| \boldsymbol{w}^{(k)} \right\|_2^2$$

Independent unmixings problems

To structured data…

Gromov-Wasserstein dictionary learning



Step 1. Linear modeling of a graph $\mathcal{G} = (\mathbf{C}, \mathbf{h})$

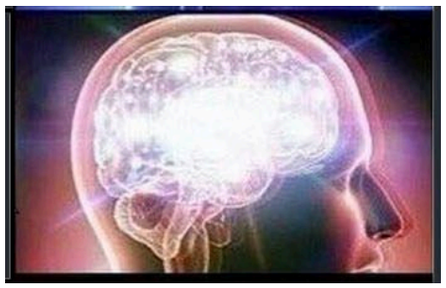
$$\min_{\boldsymbol{w} \in \Sigma_d} GW_2^2 \left(\boldsymbol{C}, \sum_{i=1}^d w_i \overline{\boldsymbol{C}}_i \right) - \lambda \|\boldsymbol{w}\|_2^2$$

Result: the GW distance in the embedding space has a proxy

$$GW_2 \left(\sum_{i=1}^d w_i^{(1)} \overline{\boldsymbol{C}}_i, \sum_{i=1}^d w_i^{(2)} \overline{\boldsymbol{C}}_i \right) \leq \left\| \boldsymbol{w}^{(1)} - \boldsymbol{w}^{(2)} \right\|_{\mathbf{M}}$$

To structured data…

Gromov-Wasserstein dictionary learning



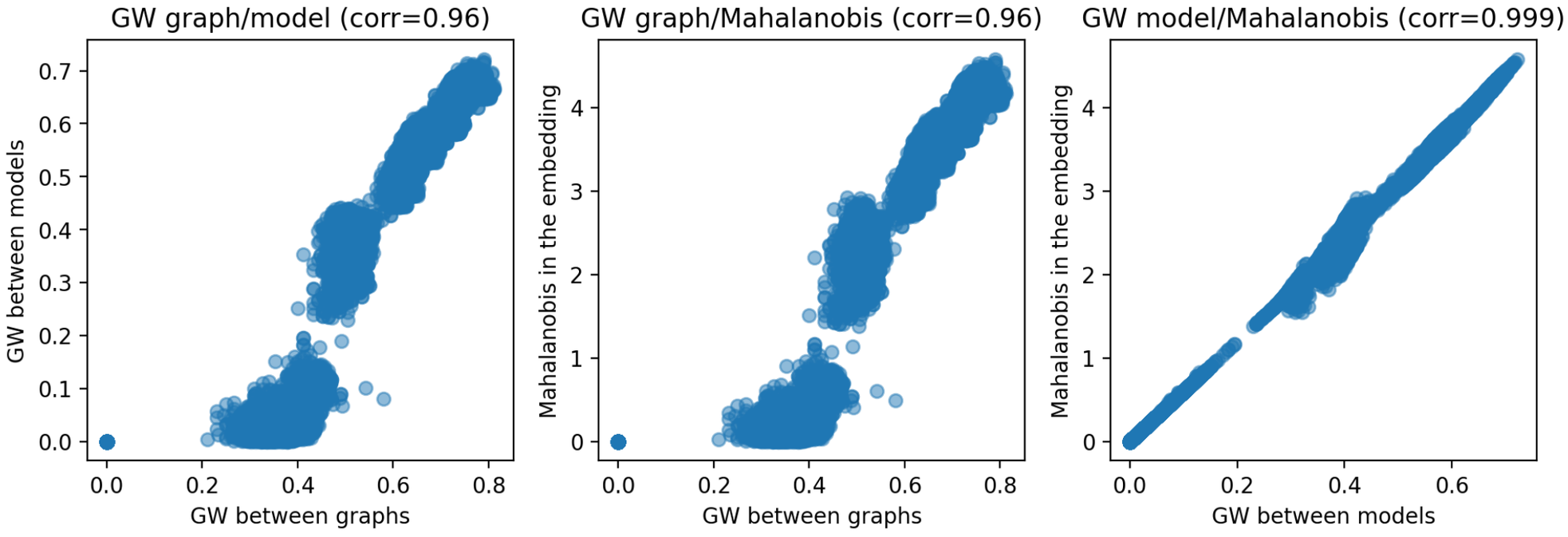
Step 1. Linear modeling of a graph $\mathcal{G} = (\mathbf{C}, \mathbf{h})$

$$\min_{\boldsymbol{w} \in \Sigma_d} GW_2^2 \left(\boldsymbol{C}, \sum_{i=1}^d w_i \overline{\boldsymbol{C}}_i \right) - \lambda \|\boldsymbol{w}\|_2^2$$

Result: the GW distance in the embedding space has a proxy

$$GW_2 \left(\sum_{i=1}^d w_i^{(1)} \overline{\boldsymbol{C}}_i, \sum_{i=1}^d w_i^{(2)} \overline{\boldsymbol{C}}_i \right) \leq \left\| \boldsymbol{w}^{(1)} - \boldsymbol{w}^{(2)} \right\|_{\mathbf{M}}$$

Fast proxy for GW!
-> can be plugged into any learning algo with kernel



To structured data…

Gromov-Wasserstein dictionary learning

Step 1. Linear modeling of a graph $\mathcal{G} = (\mathbf{C}, \mathbf{h})$

$$\min_{\boldsymbol{w} \in \Sigma_d} GW_2^2 \left(\boldsymbol{C}, \sum_{i=1}^d w_i \overline{\boldsymbol{C}}_i \right) - \lambda \|\boldsymbol{w}\|_2^2$$

Step 2. Linear modeling of a bunch of graphs $\{\mathcal{G}^k : (\mathbf{C}^{(k)}, \mathbf{h}^{(k)})\}_{k \in [\![K]\!]}$

$$\min_{\left\{\boldsymbol{w}^{(k)}\right\}_{k \in [K]}} \sum_{k=1}^K GW_2^2 \left(\boldsymbol{C}^{(k)}, \sum_{i=1}^d w_i^{(k)} \overline{\boldsymbol{C}}_i \right) - \lambda \left\|\boldsymbol{w}^{(k)}\right\|_2^2$$

Step 3. Learn the dictionary

$$\min_{\left\{\boldsymbol{w}^{(k)}\right\}_{k \in [K]}, \left\{\overline{\mathbf{C}_i}^{(i)}\right\}_{i \in [d]}} \sum_{k=1}^K GW_2^2 \left(\boldsymbol{C}^{(k)}, \sum_{i=1}^d w_i^{(k)} \overline{\boldsymbol{C}}_i \right) - \lambda \left\|\boldsymbol{w}^{(k)}\right\|_2^2$$

-> Alternate minimization + stochastic approach

To structured data…

Gromov-Wasserstein dictionary learning

Alternate minimization:

$$\min_{\left\{\boldsymbol{w}^{(k)}\right\}_{k\in[K]},\left\{\overline{\mathbf{C}_i}^{(i)}\right\}_{i\in[d]}} \sum_{k=1}^{K} GW_2^2\left(\boldsymbol{C}^{(k)}, \sum_{i=1}^{d} w_i^{(k)}\overline{\boldsymbol{C}}_i\right) - \lambda\left\|\boldsymbol{w}^{(k)}\right\|_2^2$$

Algorithm GDL

repeat

1: Sample a minibatch of graphs $\mathcal{B}$

2: Compute optimal $\{\mathbf{w}^{(k)}, \mathbf{T}^{(k)}\}_{k\in\mathcal{B}}$ with the unmixing algo

3: Projected gradient step with estimated gradients $\widetilde{\nabla}_{\overline{\boldsymbol{C}}_i}, \forall i \in [d] : \overline{\boldsymbol{C}}_i \leftarrow \mathrm{Proj}_{S_N(\mathbb{R})}\left(\overline{\boldsymbol{C}}_i - \eta_C \widetilde{\nabla}_{\overline{\boldsymbol{C}}_i}\right)$

until convergence

Complexity

$O(M^2)$

M: higher order of the graph in the dataset

To structured data…

Gromov-Wasserstein dictionary learning

Step 4. Learn the weights of the graph representation !!!

$$\min_{\left\{\left(\boldsymbol{w}^{(k)},\boldsymbol{v}^{(k)}\right)\right\}_{k\in[K]},\left\{\left(\overline{\boldsymbol{C}}_i,\bar{\mathbf{h}}_i\right)\right\}_{i\in[d]}} \sum_{k=1}^{K} GW_2^2\left(\boldsymbol{C}^{(k)},\widetilde{\boldsymbol{C}}\left(\boldsymbol{w}^{(k)}\right),\boldsymbol{h}^{(k)},\widetilde{\boldsymbol{h}}\left(\boldsymbol{v}^{(k)}\right)\right)$$
$$- \lambda\left\|\boldsymbol{w}^{(k)}\right\|_2^2 - \mu\left\|\boldsymbol{v}^{(k)}\right\|_2^2$$

$$\widetilde{\boldsymbol{h}}\left(\boldsymbol{v}^{(k)}\right) = \sum_{i=1}^{d} v_i^{(k)}\overline{\boldsymbol{h}}_i, \quad \widetilde{\boldsymbol{C}}\left(\boldsymbol{w}^{(k)}\right) = \sum_{i=1}^{d} w_i^{(k)}\overline{\boldsymbol{C}}_i$$

To structured data…

Gromov-Wasserstein dictionary learning

Step 4. Learn the weights of the graph representation !!!

$$\min_{\left\{\left(\boldsymbol{w}^{(k)},\boldsymbol{v}^{(k)}\right)\right\}_{k\in[K]},\left\{\left(\overline{\boldsymbol{C}}_i,\bar{\mathbf{h}}_i\right)\right\}_{i\in[d]}} \sum_{k=1}^{K} GW_2^2\left(\boldsymbol{C}^{(k)},\widetilde{\boldsymbol{C}}\left(\boldsymbol{w}^{(k)}\right),\boldsymbol{h}^{(k)},\widetilde{\boldsymbol{h}}\left(\boldsymbol{v}^{(k)}\right)\right)$$
$$-\lambda\left\|\boldsymbol{w}^{(k)}\right\|_2^2-\mu\left\|\boldsymbol{v}^{(k)}\right\|_2^2$$

Linear representation of the structure

$$\widetilde{\boldsymbol{h}}\left(\boldsymbol{v}^{(k)}\right)=\sum_{i=1}^{d} v_i^{(k)}\overline{\boldsymbol{h}}_i,\quad \widetilde{\boldsymbol{C}}\left(\boldsymbol{w}^{(k)}\right)=\sum_{i=1}^{d} w_i^{(k)}\overline{\boldsymbol{C}}_i$$

To structured data…

Gromov-Wasserstein dictionary learning

Step 4. Learn the weights of the graph representation !!!

$$\min_{\left\{\left(\boldsymbol{w}^{(k)},\boldsymbol{v}^{(k)}\right)\right\}_{k\in[K]},\left\{\left(\overline{\boldsymbol{C}}_i,\bar{\mathbf{h}}_i\right)\right\}_{i\in[d]}} \sum_{k=1}^{K} GW_2^2\left(\boldsymbol{C}^{(k)},\widetilde{\boldsymbol{C}}\left(\boldsymbol{w}^{(k)}\right),\boldsymbol{h}^{(k)},\widetilde{\boldsymbol{h}}\left(\boldsymbol{v}^{(k)}\right)\right)$$
$$-\lambda\left\|\boldsymbol{w}^{(k)}\right\|_2^2-\mu\left\|\boldsymbol{v}^{(k)}\right\|_2^2$$

Weights of the graph k

$$\widetilde{\boldsymbol{h}}\left(\boldsymbol{v}^{(k)}\right)=\sum_{i=1}^{d} v_i^{(k)}\overline{\boldsymbol{h}}_i,\quad \widetilde{\boldsymbol{C}}\left(\boldsymbol{w}^{(k)}\right)=\sum_{i=1}^{d} w_i^{(k)}\overline{\boldsymbol{C}}_i$$

To structured data…

Gromov-Wasserstein dictionary learning

Step 4. Learn the weights of the graph representation !!!

$$\left\{\left(\boldsymbol{w}^{(k)},\boldsymbol{v}^{(k)}\right)\right\}_{k\in[K]},\left\{\left(\overline{\boldsymbol{C}}_i,\bar{\mathbf{h}}_i\right)\right\}_{i\in[d]} \sum_{k=1}^{K} GW_2^2\left(\boldsymbol{C}^{(k)},\widetilde{\boldsymbol{C}}\left(\boldsymbol{w}^{(k)}\right),\boldsymbol{h}^{(k)},\widetilde{\boldsymbol{h}}\left(\boldsymbol{v}^{(k)}\right)\right)$$
$$-\lambda\left\|\boldsymbol{w}^{(k)}\right\|_2^2-\mu\left\|\boldsymbol{v}^{(k)}\right\|_2^2$$

Linear representation of the weights

$$\widetilde{\boldsymbol{h}}\left(\boldsymbol{v}^{(k)}\right)=\sum_{i=1}^{d} v_i^{(k)}\overline{\boldsymbol{h}}_i, \qquad \widetilde{\boldsymbol{C}}\left(\boldsymbol{w}^{(k)}\right)=\sum_{i=1}^{d} w_i^{(k)}\overline{\boldsymbol{C}}_i$$

To structured data…

Gromov-Wasserstein dictionary learning

Step 4. Learn the weights of the graph representation !!!

$$\left\{\left(\boldsymbol{w}^{(k)},\boldsymbol{v}^{(k)}\right)\right\}_{k\in[K]},\left\{\left(\overline{\boldsymbol{C}}_i,\bar{\mathbf{h}}_i\right)\right\}_{i\in[d]} \min \sum_{k=1}^{K} GW_2^2\left(\boldsymbol{C}^{(k)},\widetilde{\boldsymbol{C}}\left(\boldsymbol{w}^{(k)}\right),\boldsymbol{h}^{(k)},\widetilde{\boldsymbol{h}}\left(\boldsymbol{v}^{(k)}\right)\right) - \lambda\left\|\boldsymbol{w}^{(k)}\right\|_2^2 - \mu\left\|\boldsymbol{v}^{(k)}\right\|_2^2$$

Dico for the weights

$$\widetilde{\boldsymbol{h}}\left(\boldsymbol{v}^{(k)}\right)=\sum_{i=1}^{d} v_i^{(k)}\overline{\boldsymbol{h}}_i, \quad \widetilde{\boldsymbol{C}}\left(\boldsymbol{w}^{(k)}\right)=\sum_{i=1}^{d} w_i^{(k)}\overline{\boldsymbol{C}}_i$$

To structured data…

Gromov-Wasserstein dictionary learning

Step 4. Learn the weights of the graph representation !!!

$$\min_{\left\{\left(\boldsymbol{w}^{(k)},\boldsymbol{v}^{(k)}\right)\right\}_{k\in[K]},\left\{\left(\overline{\boldsymbol{C}}_i,\bar{\mathbf{h}}_i\right)\right\}_{i\in[d]}} \sum_{k=1}^{K} GW_2^2\left(\boldsymbol{C}^{(k)},\widetilde{\boldsymbol{C}}\left(\boldsymbol{w}^{(k)}\right),\boldsymbol{h}^{(k)},\widetilde{\boldsymbol{h}}\left(\boldsymbol{v}^{(k)}\right)\right) - \lambda\left\|\boldsymbol{w}^{(k)}\right\|_2^2 - \mu\left\|\boldsymbol{v}^{(k)}\right\|_2^2$$

Dico for the weights

Main result:

We have the gradient for free to learn the dico for the weights!

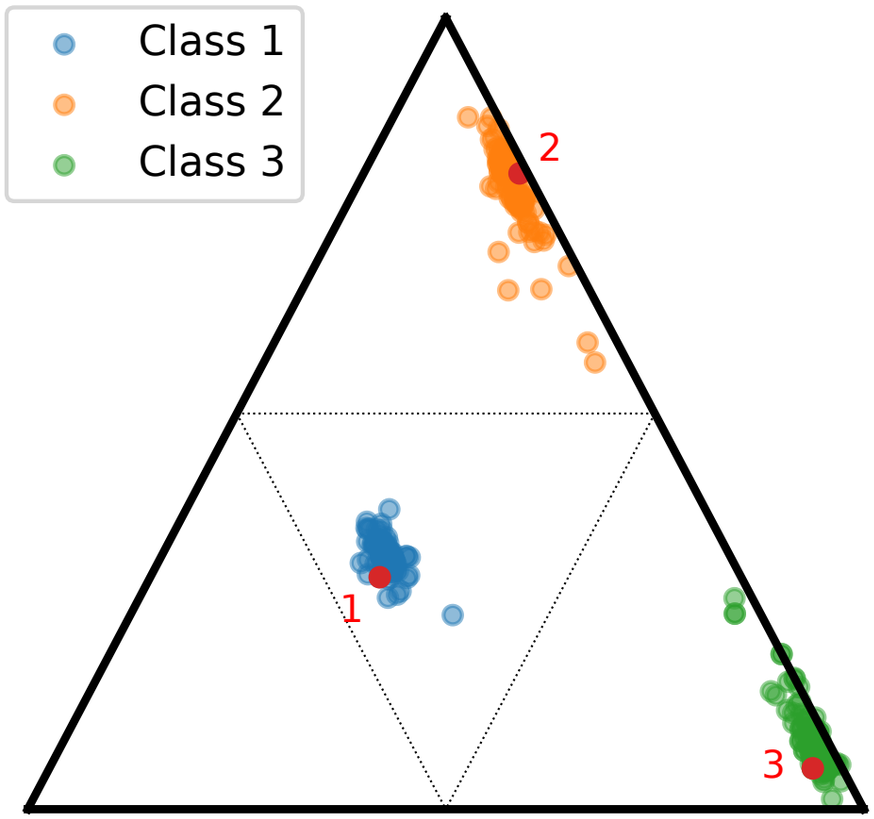
$\partial \overline{\mathbf{h}_i} GW$ is a subgradient of a linear OT problem already solved

To structured data…

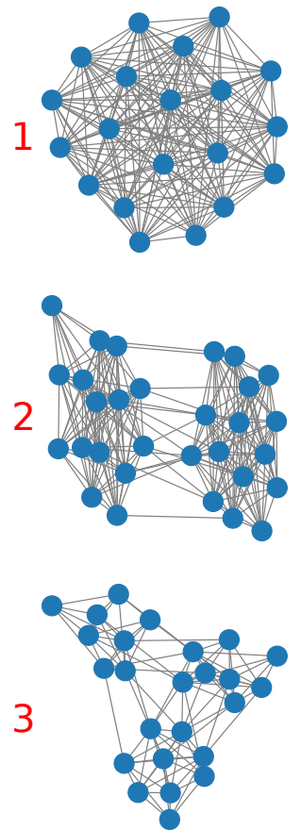
No more optim lets do stuffs…

Example: dataset SBM with communities (1,2,3)

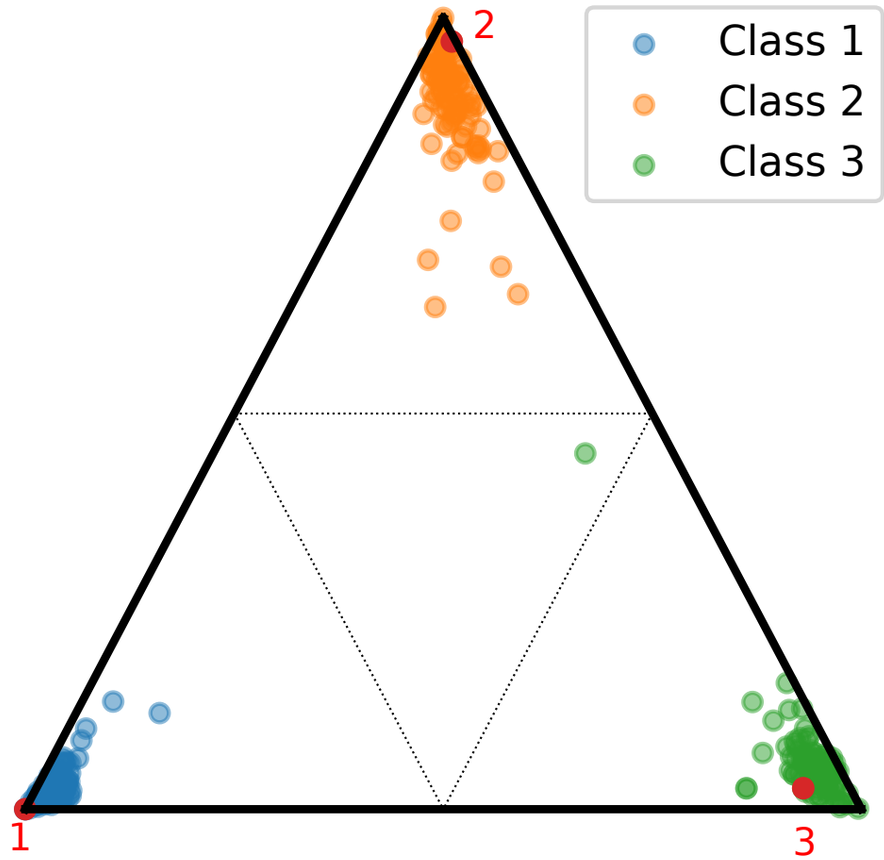
GDL unmixing $\mathbf{w}^{(k)}$ with $\lambda = 0$



Examples



GDL unmixing $\mathbf{w}^{(k)}$ with $\lambda = 0.001$

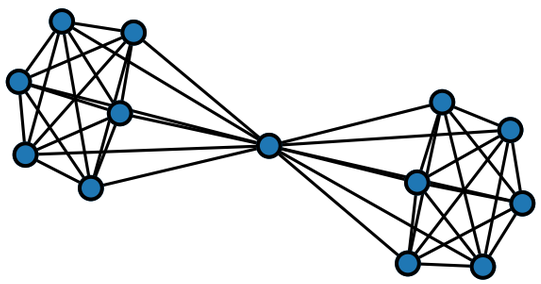


To structured data…

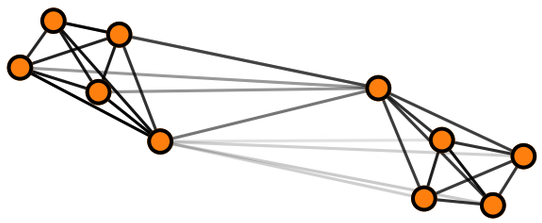
No more optim lets do stuffs…

Example: the advantage of estimating the weights

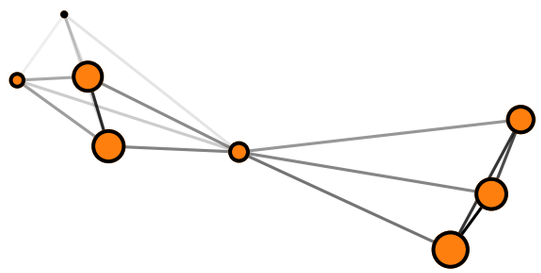
Graph from dataset



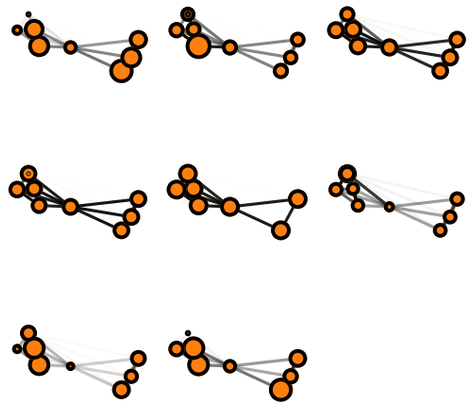
Model unif. $\mathbf{h}$ (GW=0.09)



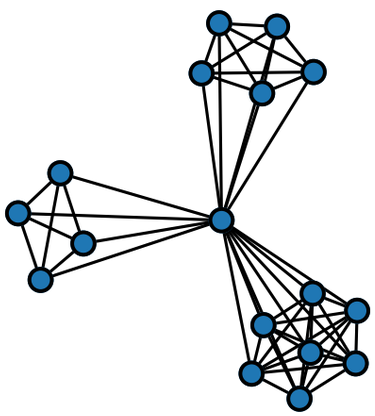
Model est. $\tilde{\mathbf{h}}$ (GW=0.08)



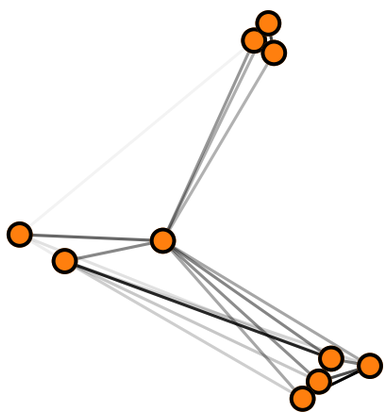
Est. $\bar{\mathbf{h}}_s$ weight atoms



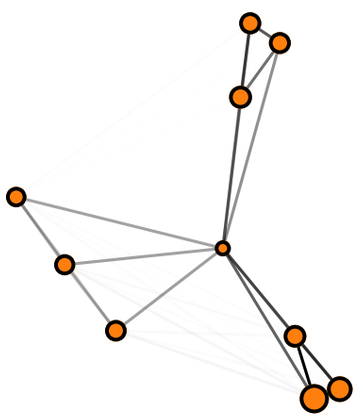
Graph from dataset



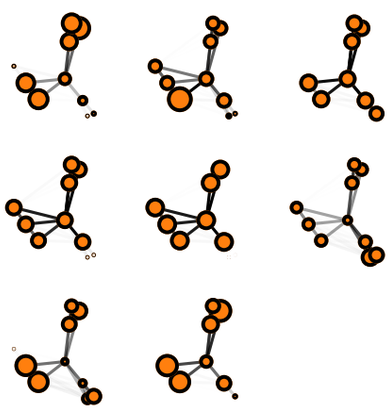
Model unif. $\mathbf{h}$ (GW=0.12)



Model est. $\tilde{\mathbf{h}}$ (GW=0.08)



Est. $\bar{\mathbf{h}}_s$ weight atoms

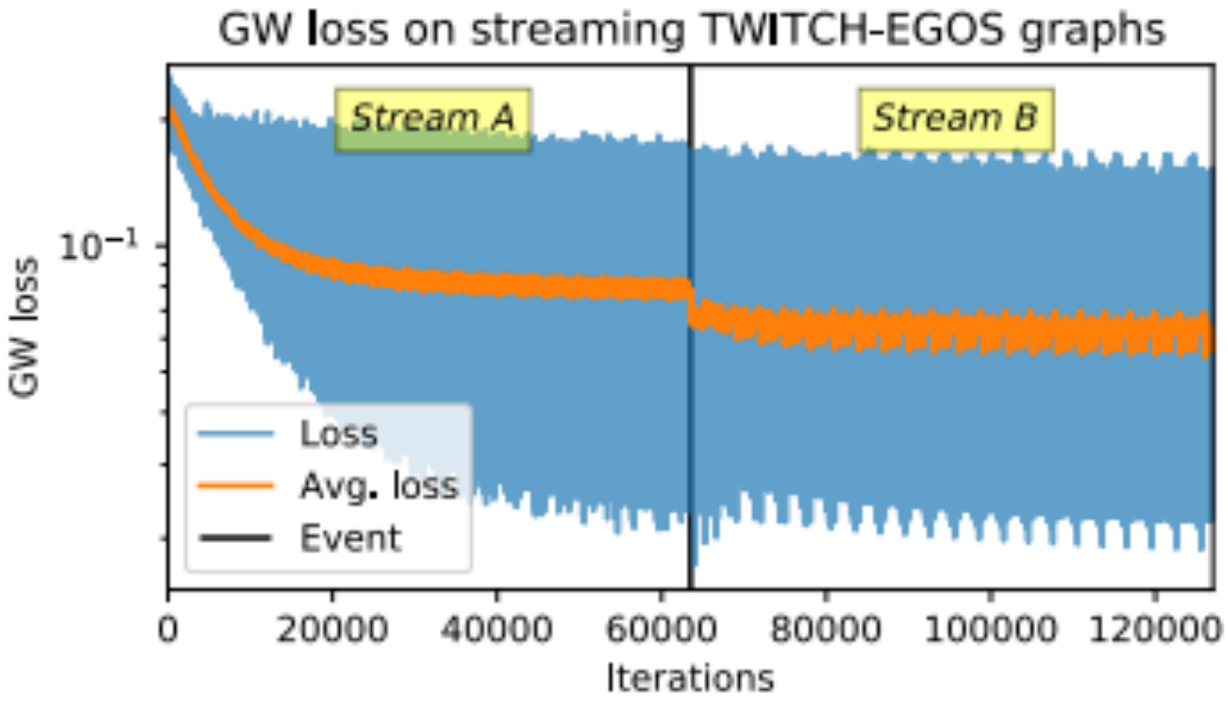


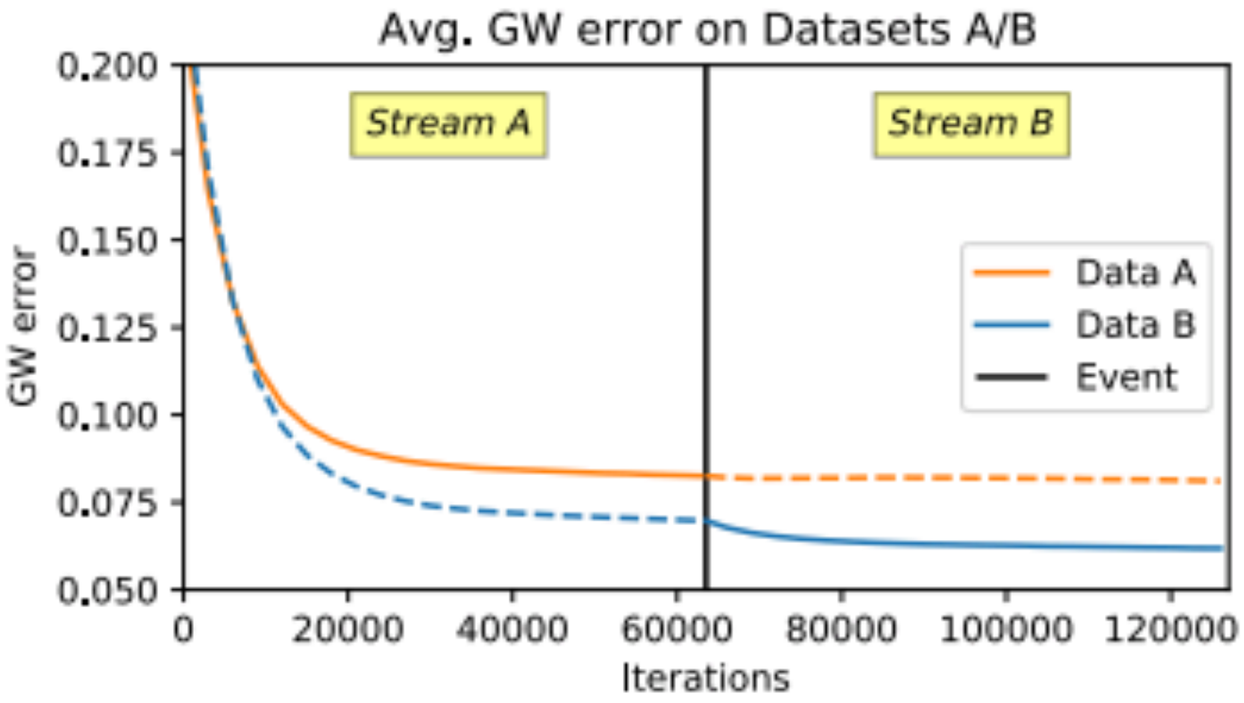
To structured data…

No more optim lets do stuffs…

Example: subspace tracking of graphs

- 2 datasets TWITCH-EGOS (60,000 graphs, classes: A,B), TRIANGLES (10,000 graphs, 10 classes)
- Graph are coming with streams
- Goal: change detection and subspace tracking



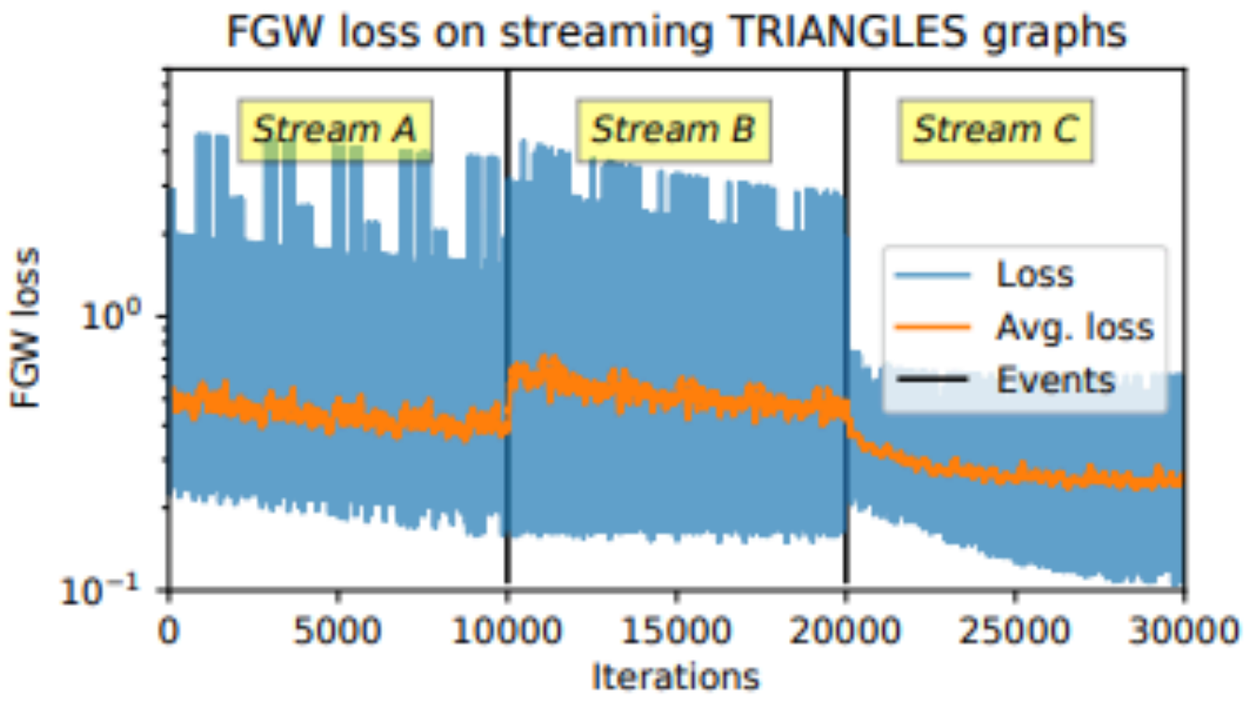


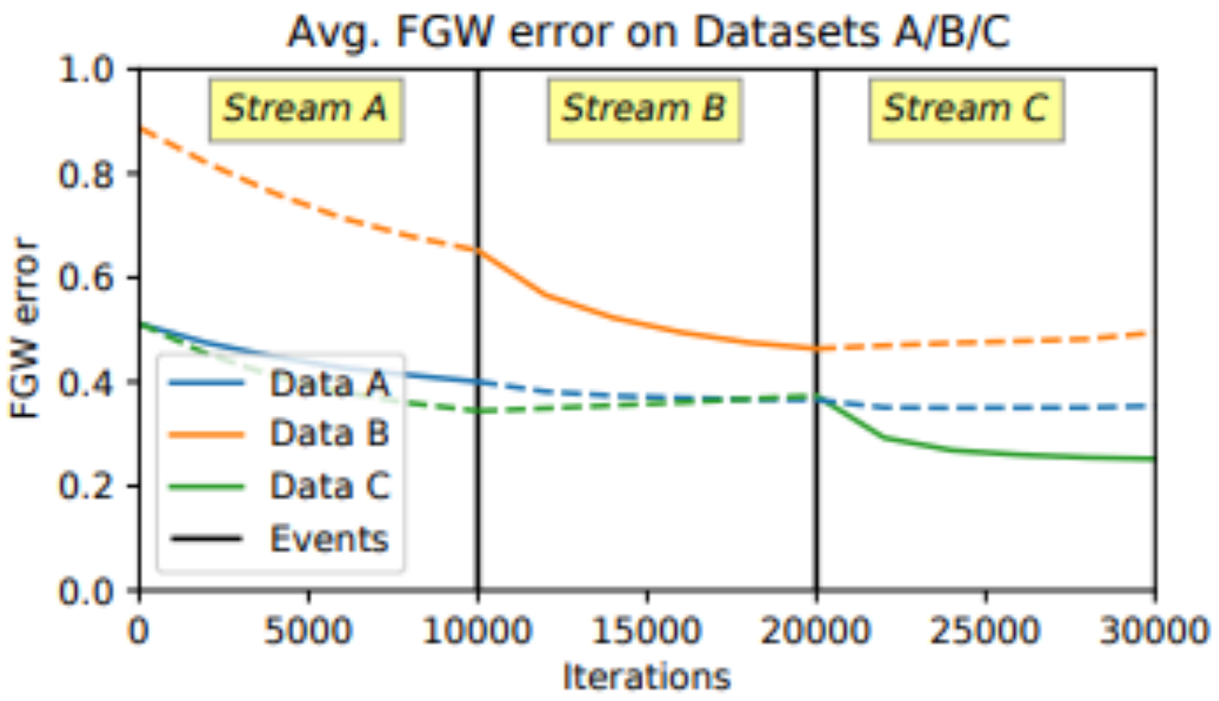
To structured data…

No more optim lets do stuffs…

Example: subspace tracking of graphs

- 2 datasets TWITCH-EGOS (60,000 graphs, classes: A,B), TRIANGLES (10,000 graphs, 10 classes)
- Graph are coming with streams
- Goal: change detecting and subspace tracking





A,B,C here = group of classes

To structured data…

No more optim lets do stuffs…

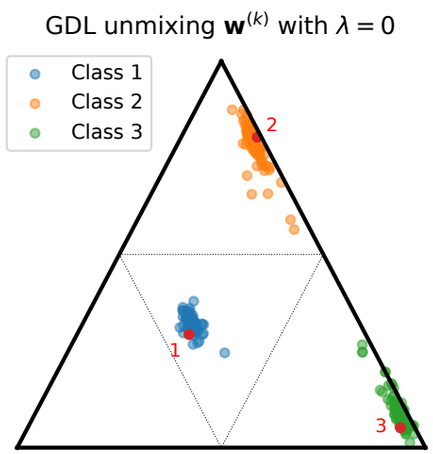
Example: K-Means with the proxy

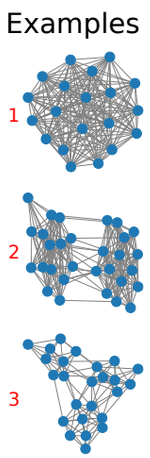
Table 1. Clustering: Rand Index computed for benchmarked approaches on real datasets.

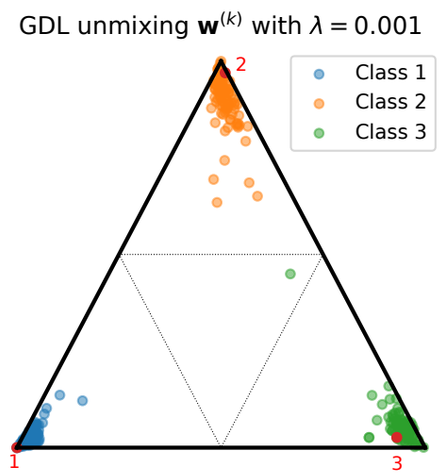
	no attribute		discrete attributes		real attributes			
models	IMDB-B	IMDB-M	MUTAG	PTC-MR	BZR	COX2	ENZYMES	PROTEIN
GDL(ours)	**51.64(0.59)**	55.41(0.20)	**70.89(0.11)**	**51.90(0.54)**	**66.42(1.96)**	**59.48(0.68)**	66.97(0.93)	**60.49(0.71)**
GWF-r	51.24 (0.02)	**55.54(0.03)**	-	-	52.42(2.48)	56.84(0.41)	**72.13(0.19)**	59.96(0.09)
GWF-f	50.47(0.34)	54.01(0.37)	-	-	51.65(2.96)	52.86(0.53)	71.64(0.31)	58.89(0.39)
GW-k	50.32(0.02)	53.65(0.07)	57.56(1.50)	50.44(0.35)	56.72(0.50)	52.48(0.12)	66.33(1.42)	50.08(0.01)
SC	50.11(0.10)	54.40(9.45)	50.82(2.71)	50.45(0.31)	42.73(7.06)	41.32(6.07)	70.74(10.60)	49.92(1.23)

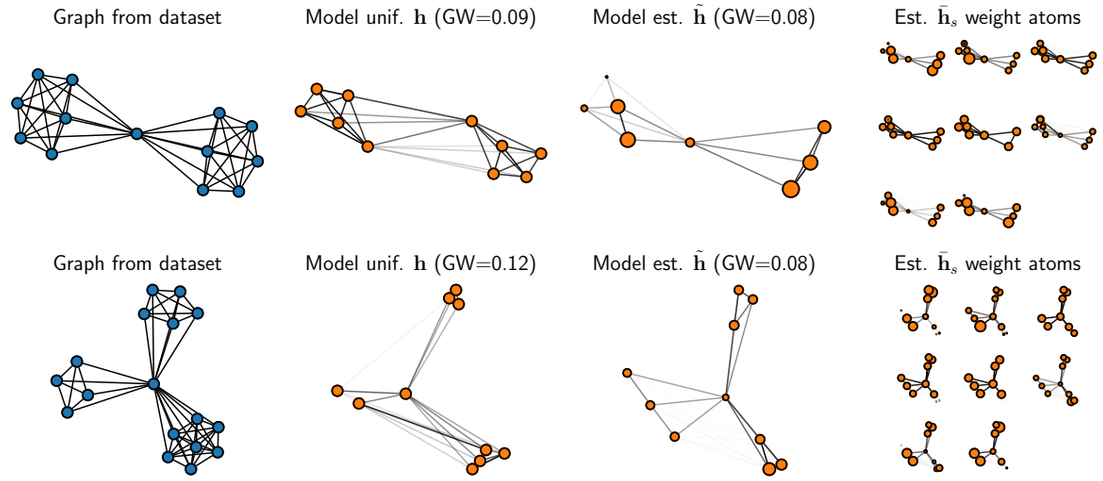
Graph dictionary learning

Conclusion









GDL

OT method for **linear Dictionary learning on graphs**

Provides interpretable **unsupervised** graphs representation

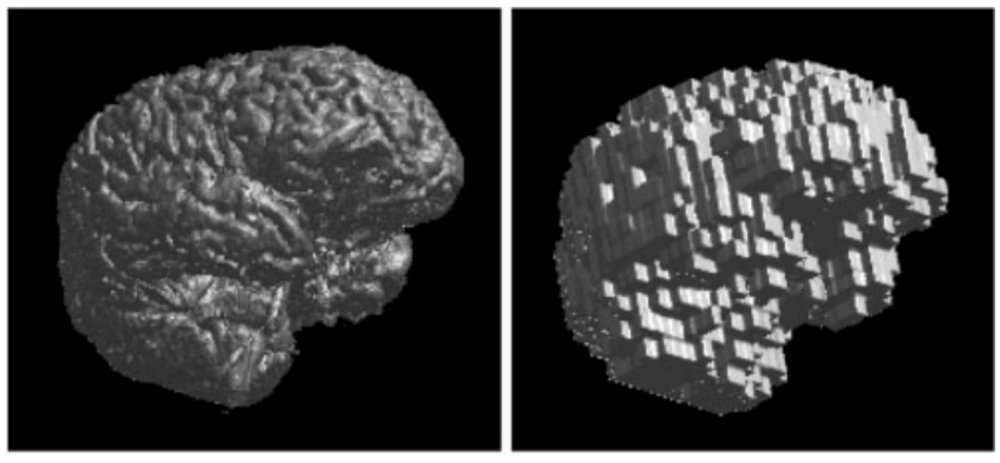
Can be used for subspace tracking, **online** change detection, compression

Perspectives

Learn the representation under Laplacian, adjacency constraint..

Graph denoising or graph inpainting

CO-Optimal Transport



CO-Optimal Transport

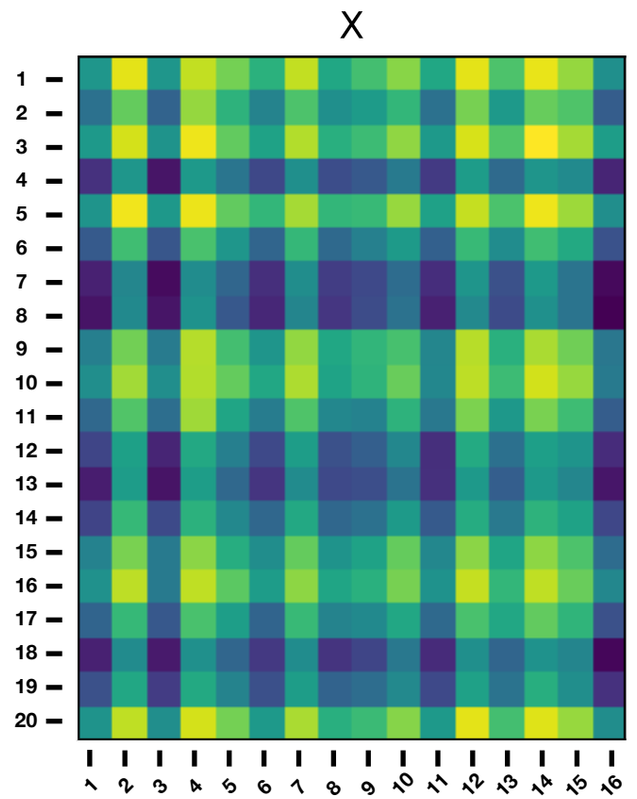
Motivations

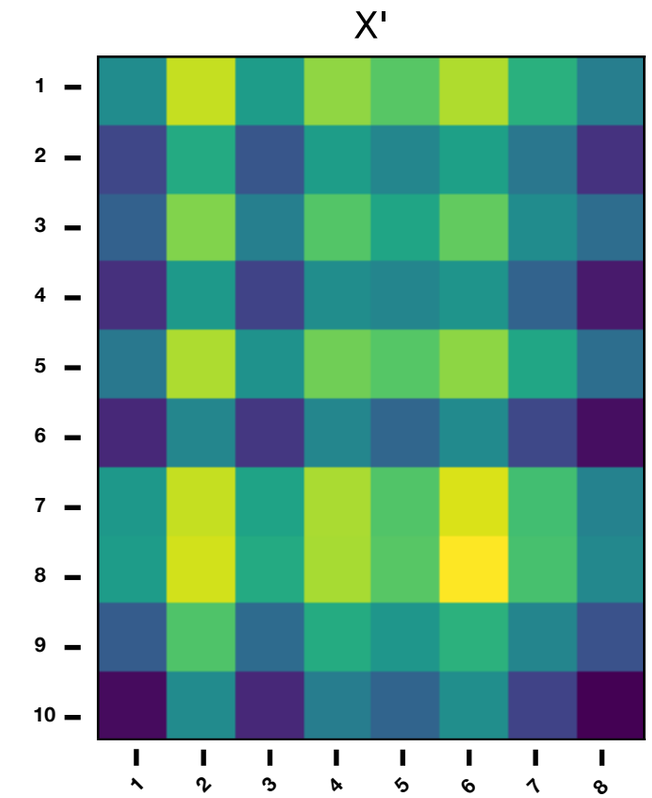
Two heterogeneous datasets

$$\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^T \in \mathbb{R}^{n \times d}$$

$$\mathbf{X}' = [\mathbf{x}'_1, \dots, \mathbf{x}'_{n'}]^T \in \mathbb{R}^{n' \times d'}$$

Row= samples, columns= features/variable





We want to measure the similarity of these two datasets (interpretable way)

Image registration [Haker 2001], HDA [Yang 2018], Word embeddings [Alvarez 2018]

GW is limited in this scenario

CO-Optimal Transport

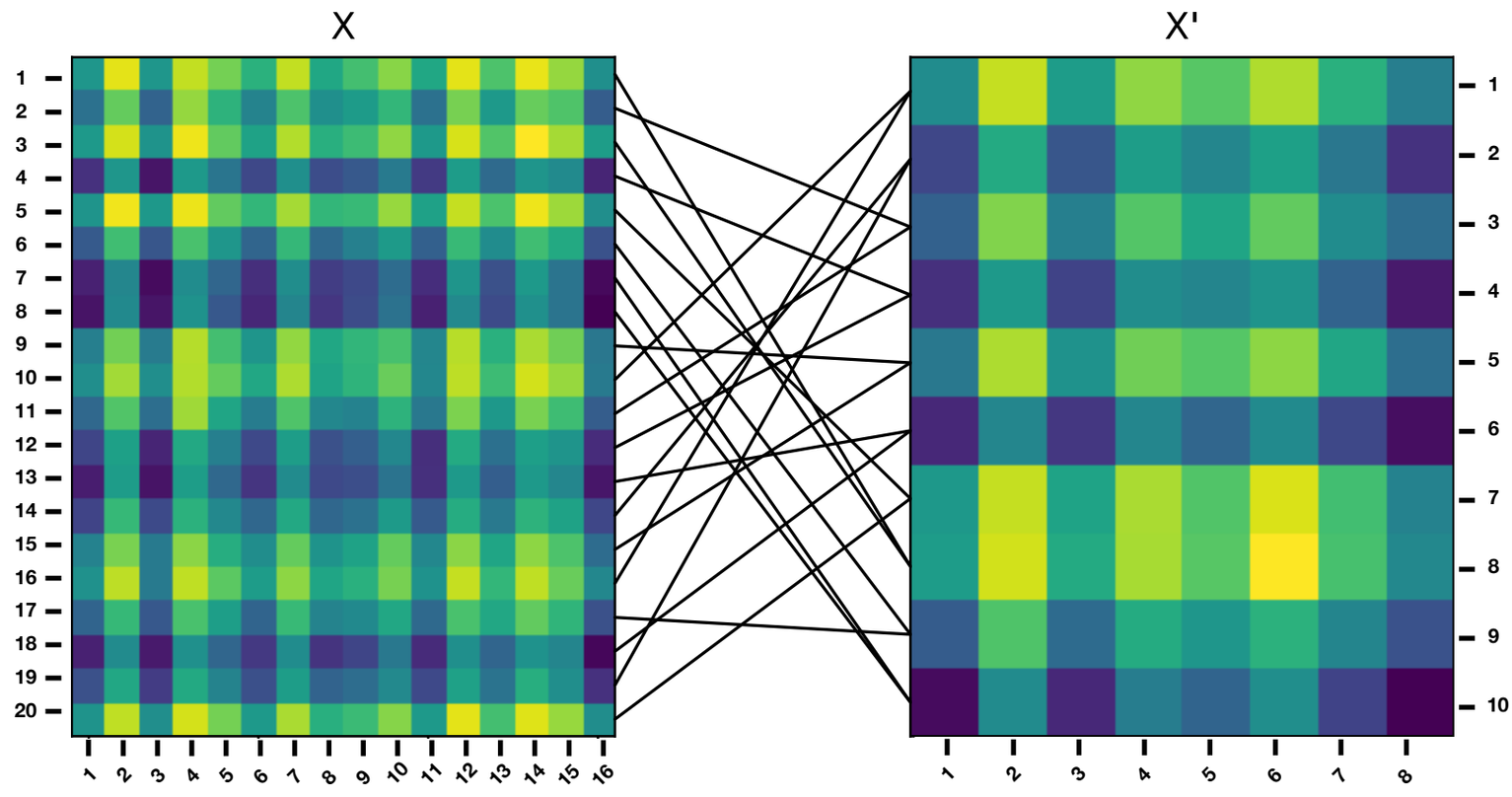
Motivations

Two heterogeneous datasets

$$\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^T \in \mathbb{R}^{n \times d}$$

$$\mathbf{X}' = [\mathbf{x}'_1, \dots, \mathbf{x}'_{n'}]^T \in \mathbb{R}^{n' \times d'}$$

Row= samples, columns= features/variable



We can apply Gromov-Wasserstein based on the pairwise distances

$$c_X(\mathbf{x}_i, \mathbf{x}_j)$$
$$c_{X'}(\mathbf{x}'_i, \mathbf{x}'_j)$$

The OT matrix gives a reordering of the samples

CO-Optimal Transport

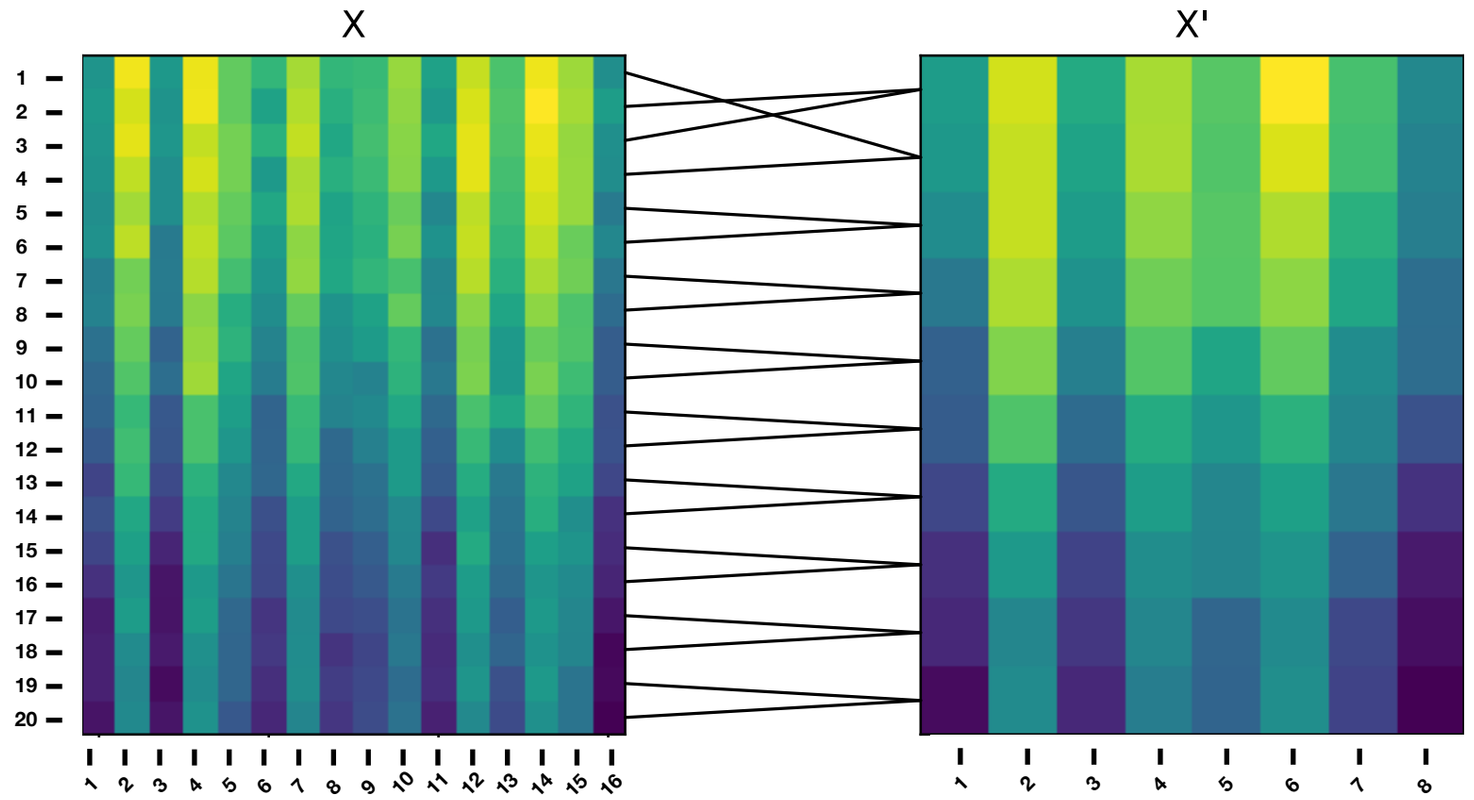
Motivations

Two heterogeneous datasets

$$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^T \in \mathbb{R}^{n \times d}$$

$$\mathbf{X}' = [\mathbf{x}'_1, \ldots, \mathbf{x}'_{n'}]^T \in \mathbb{R}^{n' \times d'}$$

Row= samples, columns= features/variable



We can apply Gromov-Wasserstein based on the pairwise distances

$$c_X(\mathbf{x}_i, \mathbf{x}_j)$$
$$c_{X'}(\mathbf{x}'_i, \mathbf{x}'_j)$$

The OT matrix gives a reordering of the samples

CO-Optimal Transport

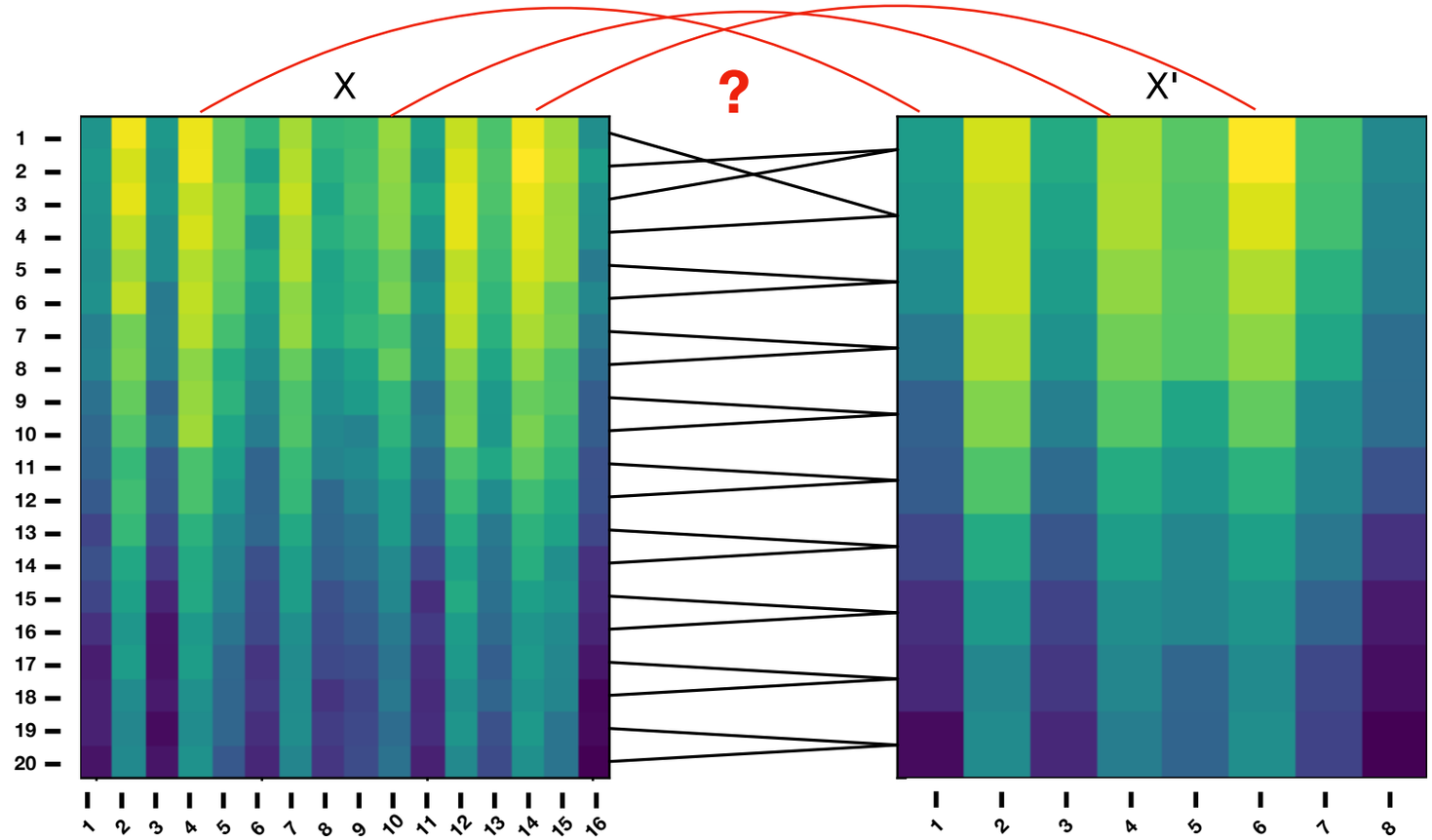
Motivations

Two heterogeneous datasets

$$\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^T \in \mathbb{R}^{n \times d}$$

$$\mathbf{X}' = [\mathbf{x}'_1, \dots, \mathbf{x}'_{n'}]^T \in \mathbb{R}^{n' \times d'}$$

Row= samples, columns= features/variable



We can apply Gromov-Wasserstein based on the pairwise distances

$$c_X(\mathbf{x}_i, \mathbf{x}_j)$$
$$c_{X'}(\mathbf{x}'_i, \mathbf{x}'_j)$$

The OT matrix gives a reordering of the samples

But discards the relationship between the features…

CO-Optimal Transport

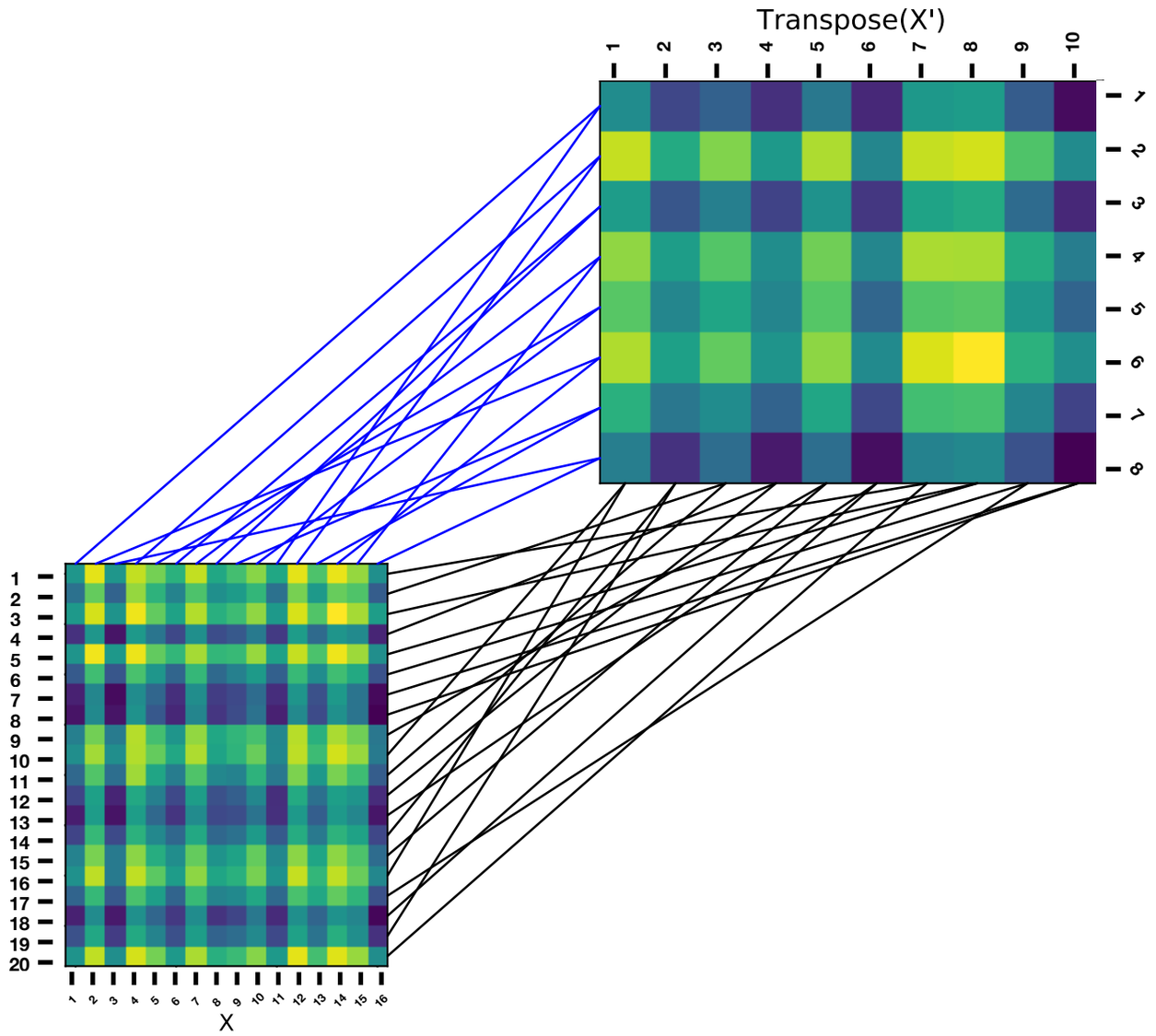
Motivations

Two heterogeneous datasets

$$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^T \in \mathbb{R}^{n \times d}$$

$$\mathbf{X}' = [\mathbf{x}'_1, \ldots, \mathbf{x}'_{n'}]^T \in \mathbb{R}^{n' \times d'}$$

Row= samples, columns= features/variable



COOT: estimate a transport matrix between the samples **and** one between the features

These matrices are **estimated jointly** and can be used for interpreting relationships across spaces

CO-Optimal Transport

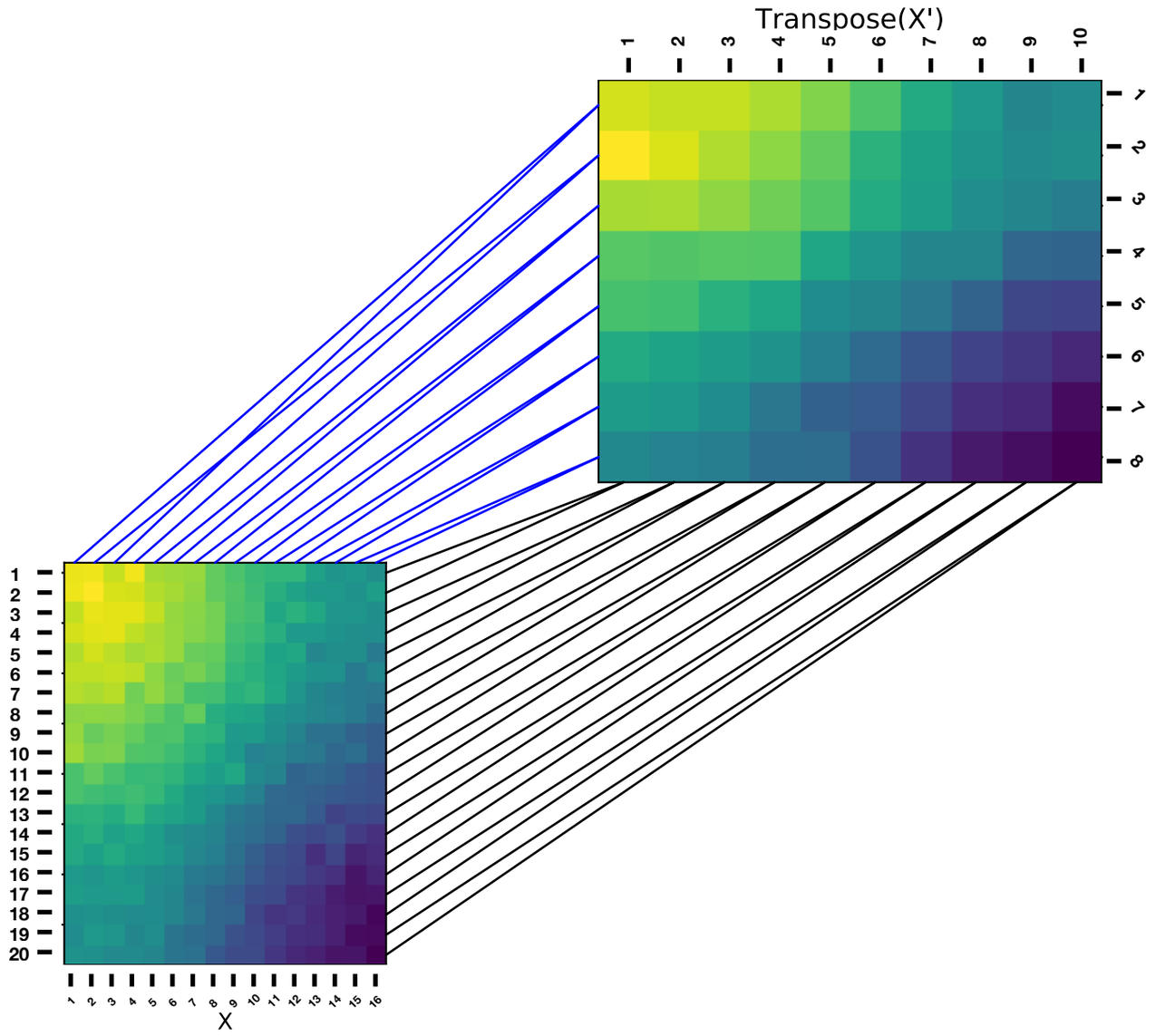
Motivations

Two heterogeneous datasets

$$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^T \in \mathbb{R}^{n \times d}$$

$$\mathbf{X}' = [\mathbf{x}'_1, \ldots, \mathbf{x}'_{n'}]^T \in \mathbb{R}^{n' \times d'}$$

Row= samples, columns= features/variable



COOT: estimate a transport matrix between the samples **and** one between the features

These matrices are **estimated jointly** and can be used for interpreting relationships across spaces

CO-Optimal Transport

Formulation & example

Two heterogeneous datasets

$$\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^T \in \mathbb{R}^{n \times d}$$

$$\mathbf{X}' = [\mathbf{x}'_1, \dots, \mathbf{x}'_{n'}]^T \in \mathbb{R}^{n' \times d'}$$

Weights

Samples: $\mathbf{w} \in \Sigma_n, \mathbf{w}' \in \Sigma_{n'}$

Features: $\mathbf{v} \in \Sigma_d, \mathbf{v}' \in \Sigma_{d'}$

CO-Optimal Transport

$$\min_{\substack{\boldsymbol{\pi}^s \in \Pi(\mathbf{w}, \mathbf{w}') \\ \boldsymbol{\pi}^v \in \Pi(\mathbf{v}, \mathbf{v}')}} \sum_{i,j,k,l} |X_{i,k} - X'_{j,l}|^p \boldsymbol{\pi}^s_{i,j} \boldsymbol{\pi}^v_{k,l}$$

$\boldsymbol{\pi}^s$: transport matrix between the samples

$\boldsymbol{\pi}^v$: transport matrix between the features/variables

CO-Optimal Transport

Formulation & example

Two heterogeneous datasets

$$\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^T \in \mathbb{R}^{n \times d}$$

$$\mathbf{X}' = [\mathbf{x}'_1, \dots, \mathbf{x}'_{n'}]^T \in \mathbb{R}^{n' \times d'}$$

Weights

Samples: $\mathbf{w} \in \Sigma_n, \mathbf{w}' \in \Sigma_{n'}$

Features: $\mathbf{v} \in \Sigma_d, \mathbf{v}' \in \Sigma_{d'}$

CO-Optimal Transport

$$\min_{\substack{\boldsymbol{\pi}^s \in \Pi(\mathbf{w}, \mathbf{w}') \\ \boldsymbol{\pi}^v \in \Pi(\mathbf{v}, \mathbf{v}')}} \sum_{i,j,k,l} |X_{i,k} - X'_{j,l}|^p \boldsymbol{\pi}^s_{i,j} \boldsymbol{\pi}^v_{k,l}$$

Applied on the raw values

$\boldsymbol{\pi}^s$: transport matrix between the samples

$\boldsymbol{\pi}^v$: transport matrix between the features/variables

CO-Optimal Transport

Formulation & example

Two heterogeneous datasets

$$\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^T \in \mathbb{R}^{n \times d}$$

$$\mathbf{X}' = [\mathbf{x}'_1, \dots, \mathbf{x}'_{n'}]^T \in \mathbb{R}^{n' \times d'}$$

Weights

Samples: $\mathbf{w} \in \Sigma_n, \mathbf{w}' \in \Sigma_{n'}$

Features: $\mathbf{v} \in \Sigma_d, \mathbf{v}' \in \Sigma_{d'}$

CO-Optimal Transport

$$\min_{\substack{\boldsymbol{\pi}^s \in \Pi(\mathbf{w}, \mathbf{w}') \\ \boldsymbol{\pi}^v \in \Pi(\mathbf{v}, \mathbf{v}')}} \sum_{i,j,k,l} |X_{i,k} - X'_{j,l}|^p \boldsymbol{\pi}^s_{i,j} \boldsymbol{\pi}^v_{k,l}$$

Applied on the raw values

$\boldsymbol{\pi}^s$: transport matrix between the samples

$\boldsymbol{\pi}^v$: transport matrix between the features/variables

Regularized version: add an entropy term for each transport matrix

CO-Optimal Transport

Formulation & example

CO-Optimal Transport

$$\min_{\substack{\boldsymbol{\pi}^s \in \Pi(\mathbf{w}, \mathbf{w}') \\ \boldsymbol{\pi}^v \in \Pi(\mathbf{v}, \mathbf{v}')}} \sum_{i,j,k,l} |X_{i,k} - X'_{j,l}|^p \boldsymbol{\pi}^s_{i,j} \boldsymbol{\pi}^v_{k,l}$$

MNIST/USPS example:

Samples: images, Features: pixels

$n = n' = 3000$

$d = 28 * 28, d' = 16 * 16$

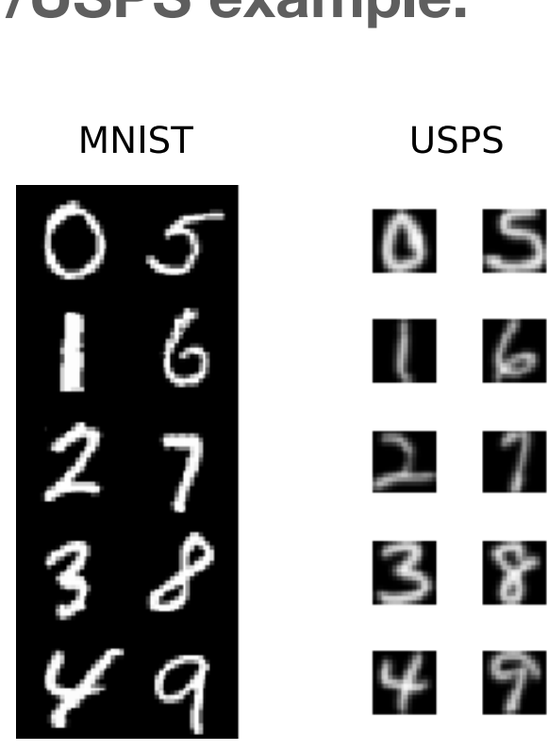
CO-Optimal Transport

Formulation & example

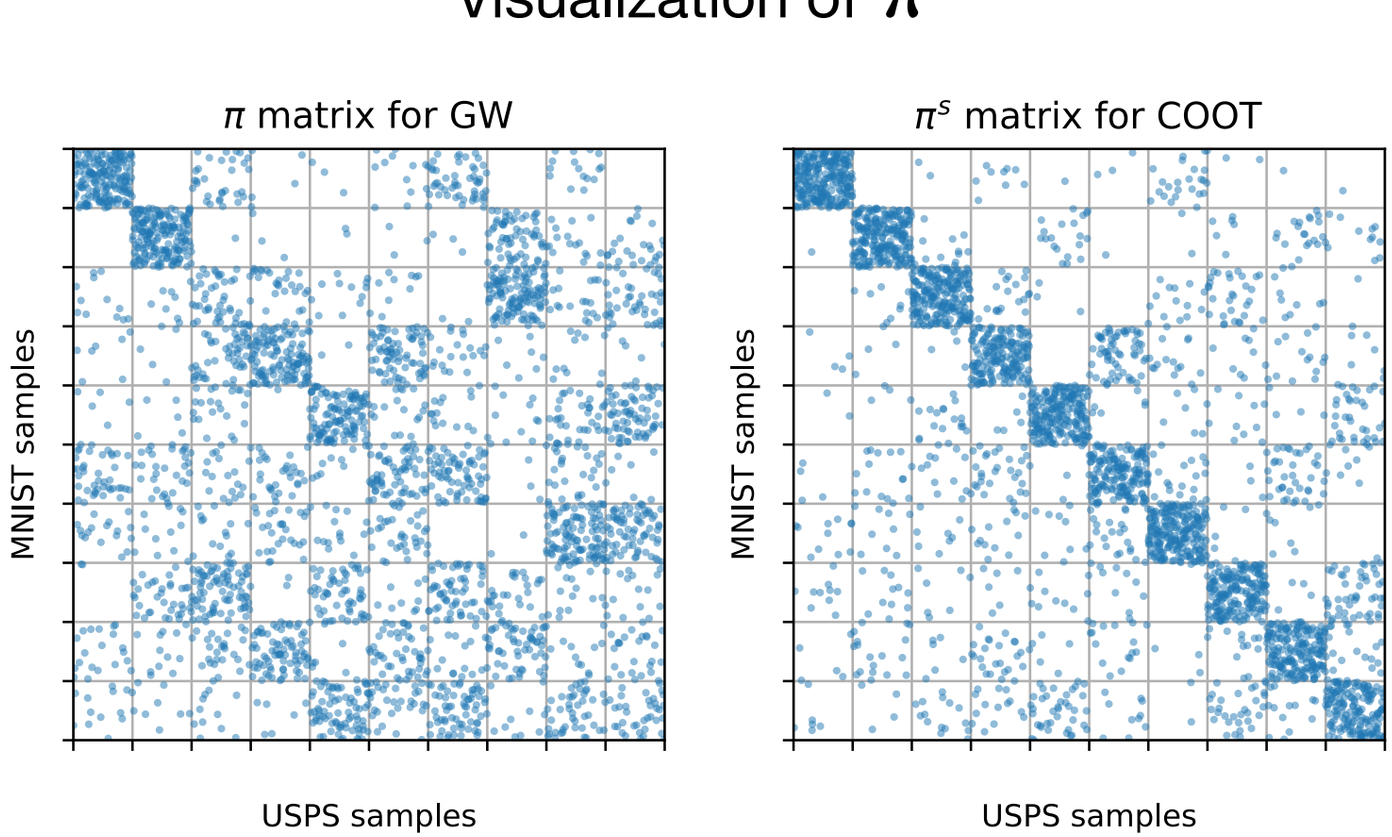
CO-Optimal Transport

$$\min_{\substack{\boldsymbol{\pi}^s \in \Pi(\mathbf{w}, \mathbf{w}') \\ \boldsymbol{\pi}^v \in \Pi(\mathbf{v}, \mathbf{v}')}} \sum_{i,j,k,l} |X_{i,k} - X'_{j,l}|^p \boldsymbol{\pi}^s_{i,j} \boldsymbol{\pi}^v_{k,l}$$

MNIST/USPS example:



Visualization of $\boldsymbol{\pi}^s$



Better class correspondence

CO-Optimal Transport

Formulation & example

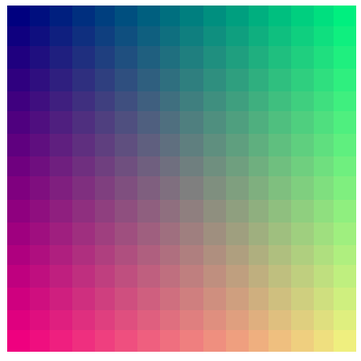
CO-Optimal Transport

$$\min_{\substack{\boldsymbol{\pi}^s \in \Pi(\mathbf{w}, \mathbf{w}') \\ \boldsymbol{\pi}^v \in \Pi(\mathbf{v}, \mathbf{v}')}} \sum_{i,j,k,l} |X_{i,k} - X'_{j,l}|^p \boldsymbol{\pi}^s_{i,j} \boldsymbol{\pi}^v_{k,l}$$

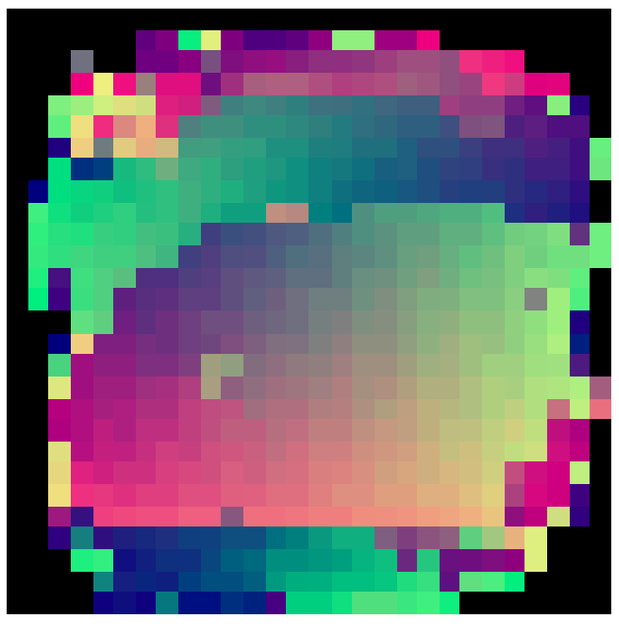
MNIST/USPS example:

Visualization of $\boldsymbol{\pi}^v$

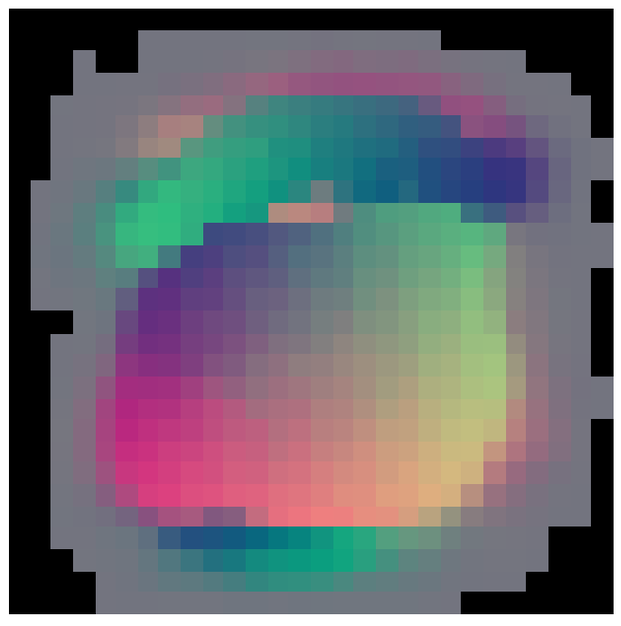
USPS colored pixels



MNIST pixels through π^v



MNIST pixels through entropic π^v



Spatial structure preserved (without supervision!)

CO-Optimal Transport

Properties

A distance w.r.t permutations of the datasets

Theorem. *COOT is a distance*

- *COOT symmetric and satisfies the triangular inequality,*

$$COOT(\mathbf{X}, \mathbf{X}'') \leq COOT(\mathbf{X}, \mathbf{X}') + COOT(\mathbf{X}', \mathbf{X}'')$$

- *Uniform weights.* $COOT(\mathbf{X}, \mathbf{X}') = 0$ *iff* $n = n', d = d'$, $\exists \sigma_1 \in S_n$ *(samples) and* $\exists \sigma_2 \in S_d$ *(features):*

$$\forall i, k \ \mathbf{X}_{i,k} = \mathbf{X}'_{\sigma_1(i), \sigma_2(k)}$$

CO-Optimal Transport

Properties

A distance w.r.t permutations of the datasets

Theorem. *COOT is a distance*

- *COOT symmetric and satisfies the triangular inequality,*

$$COOT(\mathbf{X}, \mathbf{X}'') \leq COOT(\mathbf{X}, \mathbf{X}') + COOT(\mathbf{X}', \mathbf{X}'')$$

- *Uniform weights.* $COOT(\mathbf{X}, \mathbf{X}') = 0$ *iff* $n = n', d = d'$, $\exists \sigma_1 \in S_n$ *(samples) and* $\exists \sigma_2 \in S_d$ *(features):*

$$\forall i, k \ \mathbf{X}_{i,k} = \mathbf{X}'_{\sigma_1(i),\sigma_2(k)}$$

Relation with Gromov-Wasserstein: application to relational data

$$\min_{\boldsymbol{\pi}^s, \boldsymbol{\pi}^v \in \Pi(\mathbf{w}, \mathbf{w}')} \sum_{ijkl} |C_{i,k} - C'_{j,l}|^p \pi^s_{ij} \pi^v_{kl}$$

- $\text{COOT}(\mathbf{C}, \mathbf{C}', \mathbf{w}, \mathbf{w}', \mathbf{w}, \mathbf{w}') \leq GW(\mathbf{C}, \mathbf{C}', \mathbf{w}, \mathbf{w}')$
- *When* $\mathbf{C}$ *and* $\mathbf{C}'$ *are squared Euclidean distance matrices:*

$$COOT(\mathbf{C}, \mathbf{C}', \mathbf{w}, \mathbf{w}', \mathbf{w}, \mathbf{w}') = GW(\mathbf{C}, \mathbf{C}', \mathbf{w}, \mathbf{w}')$$

and the optimal transport matrices $\boldsymbol{\pi}^{GW} = \boldsymbol{\pi}^s = \boldsymbol{\pi}^v$.

CO-Optimal Transport

Solving COOT

CO-Optimal Transport

$$\min_{\substack{\boldsymbol{\pi}^s \in \Pi(\mathbf{w}, \mathbf{w}') \\ \boldsymbol{\pi}^v \in \Pi(\mathbf{v}, \mathbf{v}')}} \sum_{i,j,k,l} |X_{i,k} - X'_{j,l}|^p \boldsymbol{\pi}^s_{i,j} \boldsymbol{\pi}^v_{k,l}$$

Non-convex bilinear program: NP-Hard

Block Coordinate Descent (BCD): alternates OT problems -> local minima [Konno 1976]

Algorithm 1 BCD for COOT

$\pi^s_{(0)} \leftarrow \mathbf{w}\mathbf{w}'^T, \pi^v_{(0)} \leftarrow \mathbf{v}\mathbf{v}'^T, k \leftarrow 0$
while $k <$ maxIt **and** $err > 0$ **do**
  $\boldsymbol{\pi}^v_{(k)} \leftarrow OT(\mathbf{v}, \mathbf{v}', \mathbf{L}(\mathbf{X}, \mathbf{X}') \otimes \boldsymbol{\pi}^s_{(k-1)})$
  $\boldsymbol{\pi}^s_{(k)} \leftarrow OT(\mathbf{w}, \mathbf{w}', \mathbf{L}(\mathbf{X}, \mathbf{X}') \otimes \boldsymbol{\pi}^v_{(k-1)})$
  $err \leftarrow ||\boldsymbol{\pi}^v_{(k-1)} - \boldsymbol{\pi}^v_{(k)}||_F$
  $k \leftarrow k + 1$
end while

CO-Optimal Transport

Solving COOT

CO-Optimal Transport

$$\min_{\substack{\boldsymbol{\pi}^s \in \Pi(\mathbf{w}, \mathbf{w}') \\ \boldsymbol{\pi}^v \in \Pi(\mathbf{v}, \mathbf{v}')}} \sum_{i,j,k,l} |X_{i,k} - X'_{j,l}|^p \boldsymbol{\pi}^s_{i,j} \boldsymbol{\pi}^v_{k,l}$$

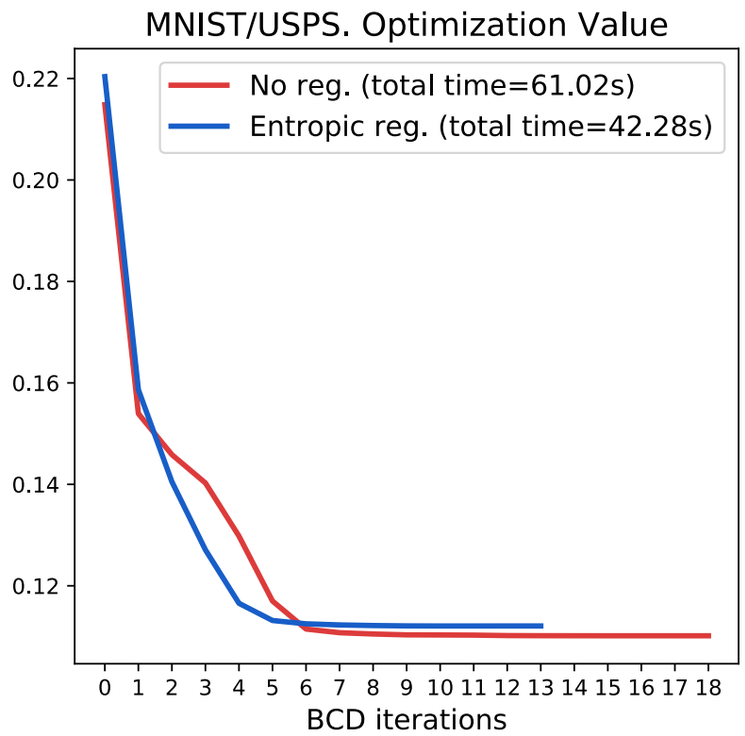
Non-convex bilinear program: NP-Hard

Block Coordinate Descent (BCD): alternates OT problems -> local minima [Konno 1976]

Algorithm 1 BCD for COOT

$\pi^s_{(0)} \leftarrow \mathbf{w}\mathbf{w}'^T, \pi^v_{(0)} \leftarrow \mathbf{v}\mathbf{v}'^T, k \leftarrow 0$
while $k <$ maxIt **and** $err > 0$ **do**
  $\boldsymbol{\pi}^v_{(k)} \leftarrow OT(\mathbf{v}, \mathbf{v}', \mathbf{L}(\mathbf{X}, \mathbf{X}') \otimes \boldsymbol{\pi}^s_{(k-1)})$
  $\boldsymbol{\pi}^s_{(k)} \leftarrow OT(\mathbf{w}, \mathbf{w}', \mathbf{L}(\mathbf{X}, \mathbf{X}') \otimes \boldsymbol{\pi}^v_{(k-1)})$
  $err \leftarrow ||\boldsymbol{\pi}^v_{(k-1)} - \boldsymbol{\pi}^v_{(k)}||_F$
  $k \leftarrow k + 1$
end while

In practice BCD converges in few iterations



CO-Optimal Transport

Solving COOT

CO-Optimal Transport

$$\min_{\substack{\boldsymbol{\pi}^s \in \Pi(\mathbf{w}, \mathbf{w}') \\ \boldsymbol{\pi}^v \in \Pi(\mathbf{v}, \mathbf{v}')}} \sum_{i,j,k,l} |X_{i,k} - X'_{j,l}|^p \boldsymbol{\pi}^s_{i,j} \boldsymbol{\pi}^v_{k,l}$$

Non-convex bilinear program: NP-Hard

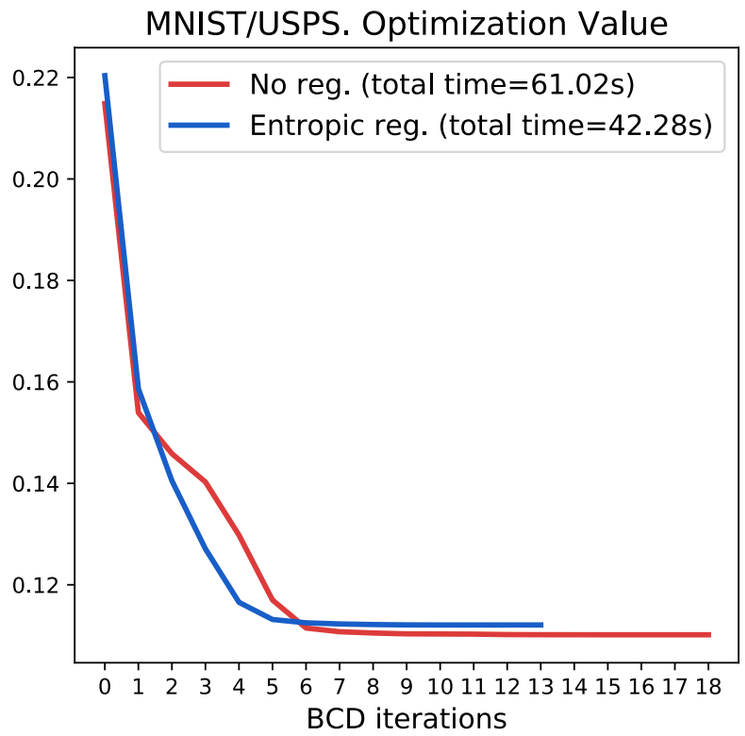
Block Coordinate Descent (BCD): alternates OT problems -> local minima [Konno 1976]

Algorithm 1 BCD for COOT

$\pi^s_{(0)} \leftarrow \mathbf{w}\mathbf{w}'^T, \pi^v_{(0)} \leftarrow \mathbf{v}\mathbf{v}'^T, k \leftarrow 0$
while $k <$ maxIt **and** $err > 0$ **do**
  $\boldsymbol{\pi}^v_{(k)} \leftarrow OT(\mathbf{v}, \mathbf{v}', \mathbf{L}(\mathbf{X}, \mathbf{X}') \otimes \boldsymbol{\pi}^s_{(k-1)})$
  $\boldsymbol{\pi}^s_{(k)} \leftarrow OT(\mathbf{w}, \mathbf{w}', \mathbf{L}(\mathbf{X}, \mathbf{X}') \otimes \boldsymbol{\pi}^v_{(k-1)})$
  $err \leftarrow ||\boldsymbol{\pi}^v_{(k-1)} - \boldsymbol{\pi}^v_{(k)}||_F$
  $k \leftarrow k + 1$
end while

can be used for GW in the concave regime!

In practice BCD converges in few iterations



CO-Optimal Transport in action

CO-Optimal Transport

Domain adaptation in a nutshell

Given a source domain with labels

$$\mathbf{X}_s = \{\mathbf{x}_i^s\}_{i=1}^{N_s}$$
$$\mathbf{Y}_s = \{\mathbf{y}_i^s\}_{i=1}^{N_s}$$

Apply/learn a classifier on

A target domain

$$\mathbf{X}_t = \{\mathbf{x}_i^t\}_{i=1}^{N_t}$$

related but
different domains..

(HDA) $\mathbf{X}_s \in \mathbb{R}^{N_s \times d}$ and $\mathbf{X}_t \in \mathbb{R}^{N_t \times d'}$ with $d \neq d'$

CO-Optimal Transport

Domain adaptation in a nutshell

Given a source domain with labels

$$\mathbf{X}_s = \{\mathbf{x}_i^s\}_{i=1}^{N_s}$$
$$\mathbf{Y}_s = \{\mathbf{y}_i^s\}_{i=1}^{N_s}$$

Apply/learn a classifier on

A target domain

$$\mathbf{X}_t = \{\mathbf{x}_i^t\}_{i=1}^{N_t}$$

related but
different domains..

$$(\text{HDA})\ \mathbf{X}_s \in \mathbb{R}^{N_s \times d} \text{ and } \mathbf{X}_t \in \mathbb{R}^{N_t \times d'} \text{ with } d \neq d'$$

Adaptation from two different embeddings
from Decaf to GoogleNet $\quad \mathbb{R}^{4096} \rightarrow \mathbb{R}^{1024}$

$$\boldsymbol{\pi}^s, \boldsymbol{\pi}^v \leftarrow \text{COOT}(\mathbf{X}_s, \mathbf{X}_t)$$

Label propagation:
$$\hat{\mathbf{Y}}_t = \boldsymbol{\pi}^s \mathbf{Y}_s$$

[Redko 2019]

Very good results on HDA

CO-Optimal Transport

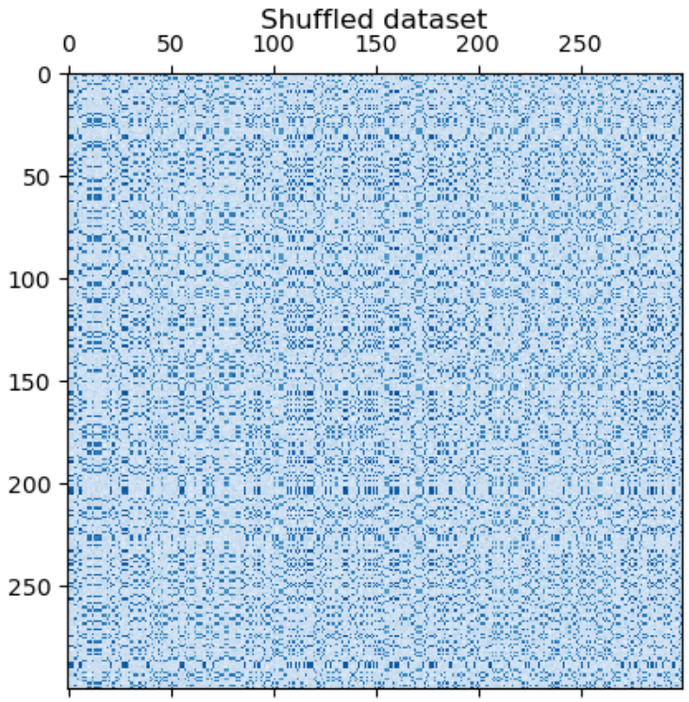
COOT in action: CO-clustering

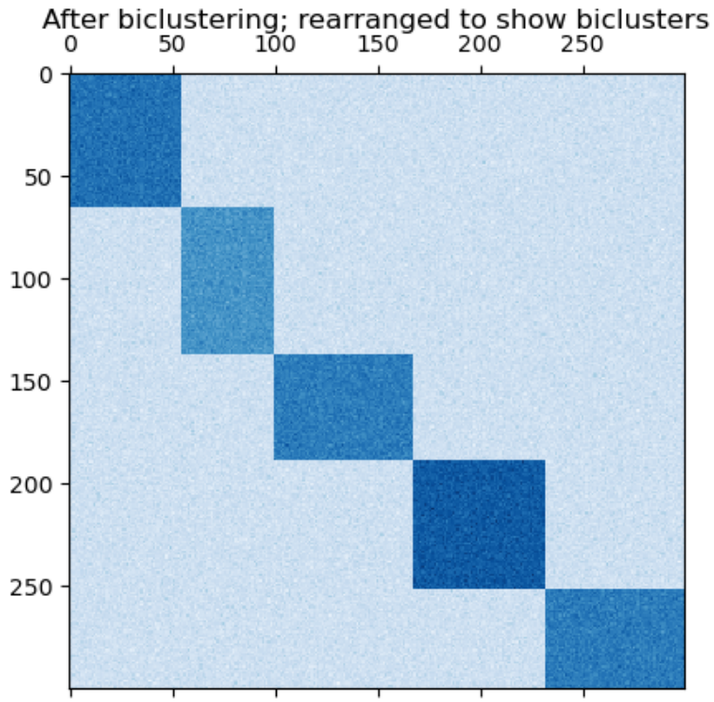
Search for a simultaneous clustering of both samples and features of a dataset

$$\mathbf{X} \in \mathbb{R}^{n \times d}$$

COOT CO-clustering

$$\min_{\mathbf{X}_c \in \mathbb{R}^{n' \times d'}} \mathrm{COOT}(\mathbf{X}, \mathbf{X}_c)$$





$\mathbf{X}_c$ with $n' < n,\ d' < d$ that summarizes $\mathbf{X}$ in the best way possible.

Solved by BCD

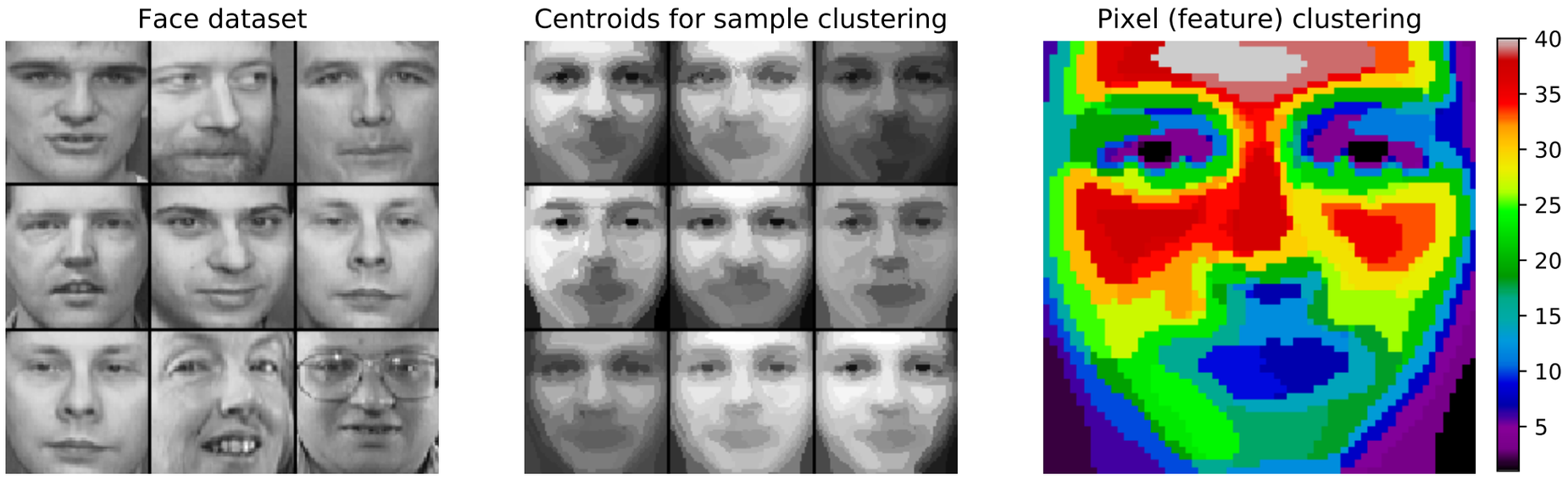
1. Obtain $\boldsymbol{\pi}^s$ and $\boldsymbol{\pi}^v$ by solving $\mathrm{COOT}(\mathbf{X}, \mathbf{X}_c)$
2. Set $\mathbf{X}_c$ to $n' d' \boldsymbol{\pi}^{s\top} \mathbf{X} \boldsymbol{\pi}^v$.

CO-Optimal Transport

COOT in action: CO-clustering

$$\min_{\mathbf{X}_c \in \mathbb{R}^{n' \times d'}} \mathrm{COOT}(\mathbf{X}, \mathbf{X}_c)$$

Olivetti faces dataset
[Samaria 1994]



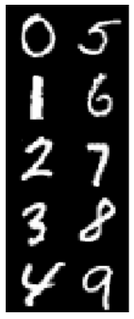
SOTA on simulated benchmarks

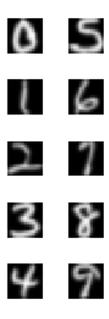
Movielens dataset (users and films)

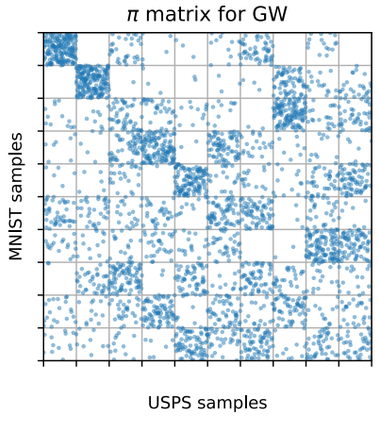
M1	M20
Shawshank Redemption (1994)	Police Story 4: Project S (Chao ji ji hua) (1993)
Schindler's List (1993)	Eye of Vichy, The (Oeil de Vichy, L') (1993)
Casablanca (1942)	Promise, The (Versprechen, Das) (1994)
Rear Window (1954)	To Cross the Rubicon (1991)
Usual Suspects, The (1995)	Daens (1992)

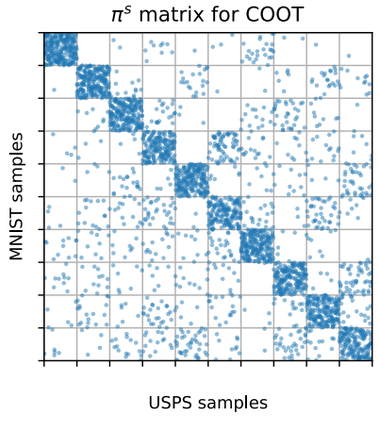
CO-Optimal Transport

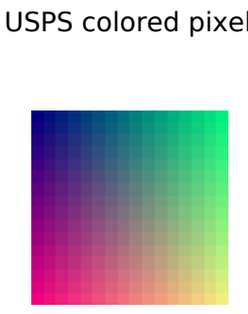
Conclusion: take away messages

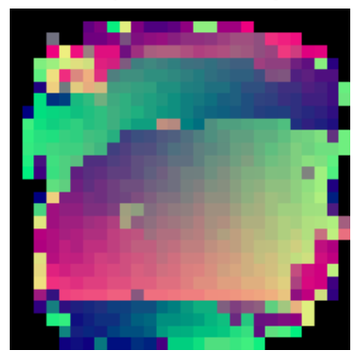


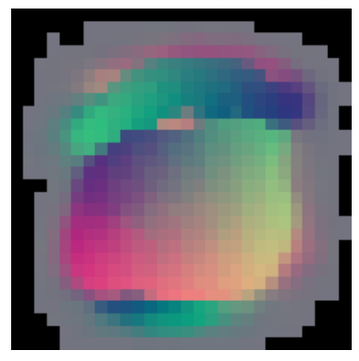












COOT

- OT method for heterogeneous dataset
- Provides interpretable correspondences between samples and features
- Works well for HDA + Can be applied for co-clustering

Future works

- Study the statistics of COOT $(n, d \to \infty \ ?)$
- Other formulations (unbalanced, extension to labeled dataset)

The POT library

Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z. Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, Léo Gautheron, Nathalie T.H. Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J. Sutherland, Romain Tavenard, Alexander Tong, Titouan Vayer; 22(78):1–8, 2021.

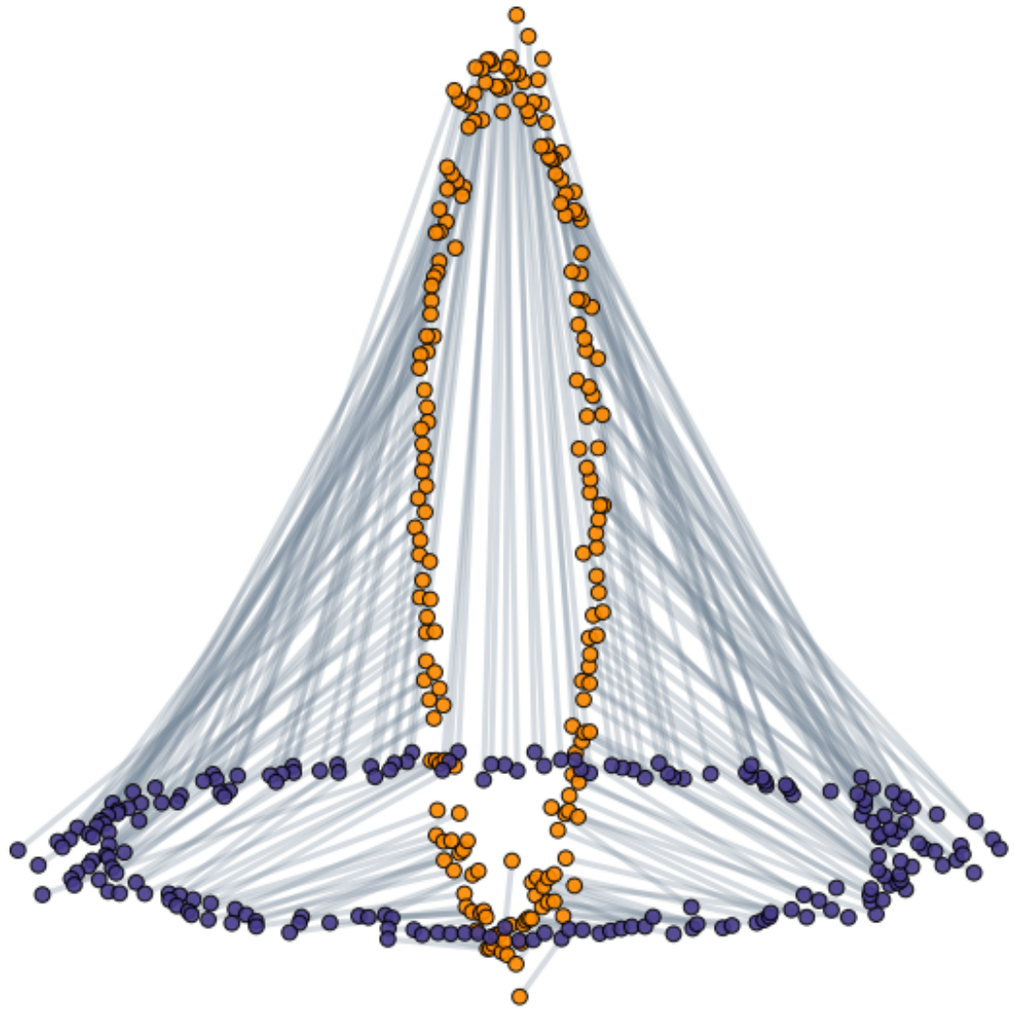
Abstract

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

Python library on Optimal Transport

- OT LP solver, Sinkhorn
- Barycenters, Domain adaptation
- Gromov, FGW, graphs OT…

Url: https://github.com/PythonOT/POT



Thank you!