A Review of Regularized Optimal Transport

Marco Cuturi

ENSAE ParisTech
École nationale de la statistique et de l'administration économique
université PARIS-SACLAY

Joint work with many people, including:
G. Peyré, A. Genevay *(ENS)*, A. Doucet *(Oxford)* J. Solomon *(MIT)*,
J.D. Benamou, N. Bonneel, F. Bach, L. Nenna *(INRIA)*,
G. Carlier (*Dauphine*).

What is Optimal Transport?

A geometric toolbox to
compare probability measures
supported on a metric space.


Monge


Kantorovich


Dantzig


Wasserstein


Brenier

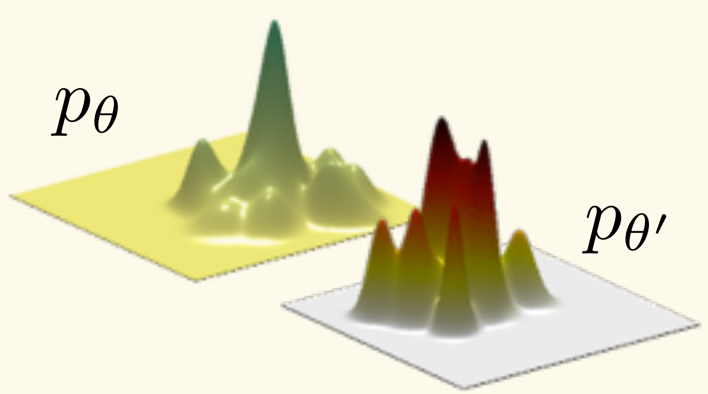

Otto


McCann


Villani

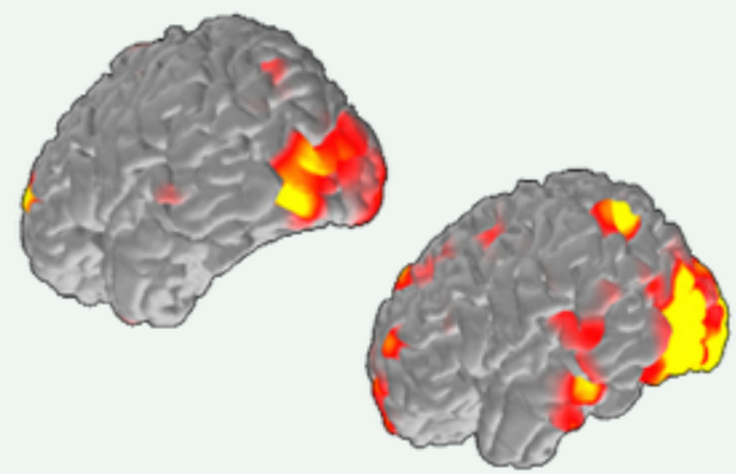
What is Optimal Transport?

A geometric toolbox to
compare **probability measures**
supported on a metric space.



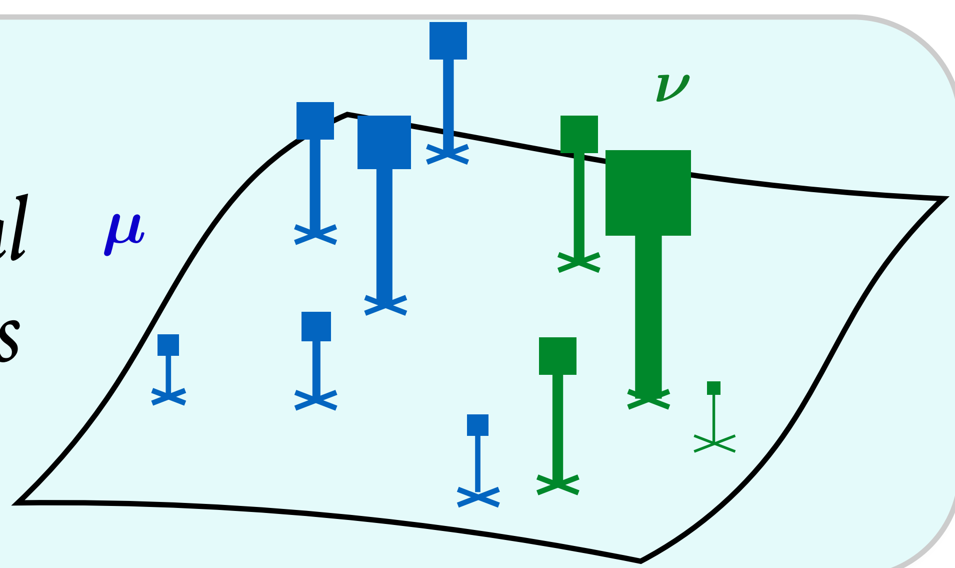
Statistical Models

Bags of features



Brain Activation Maps

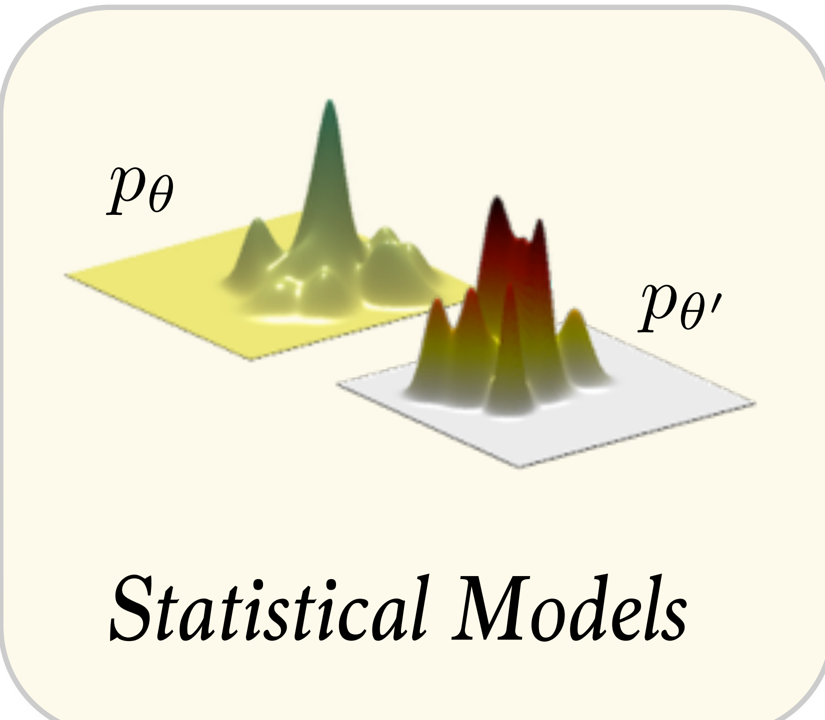
Empirical Measures

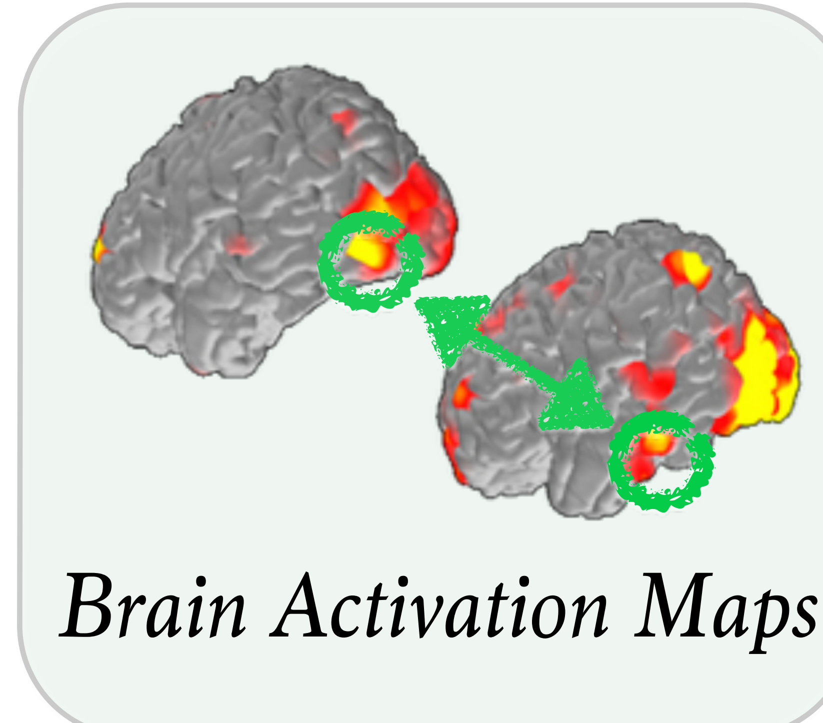


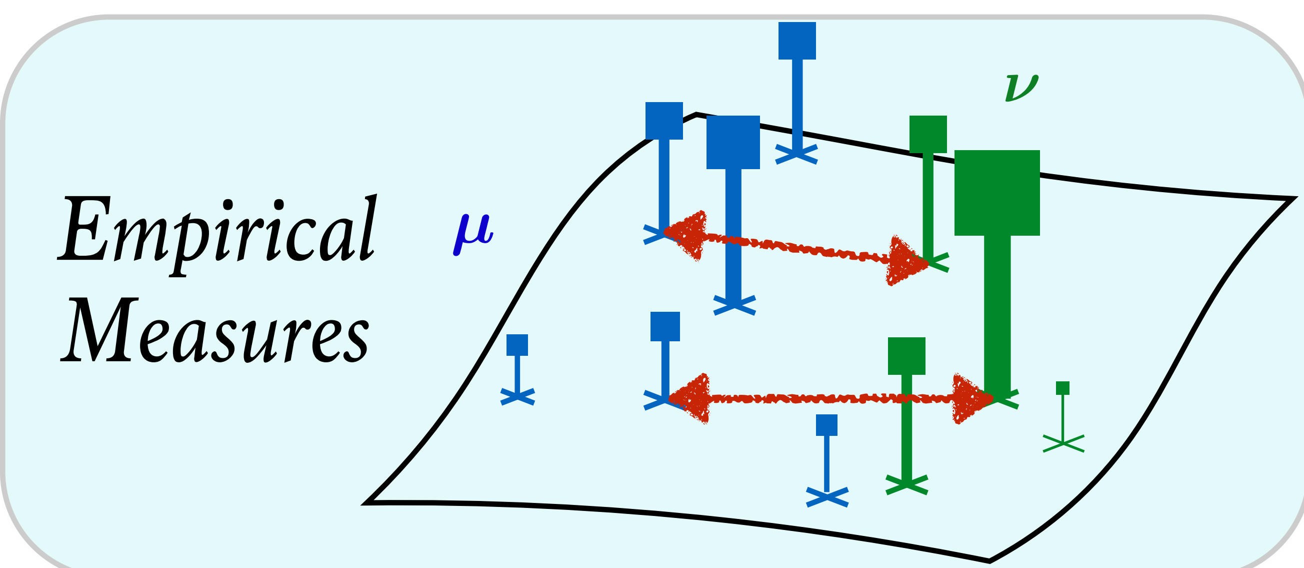
Color Histograms

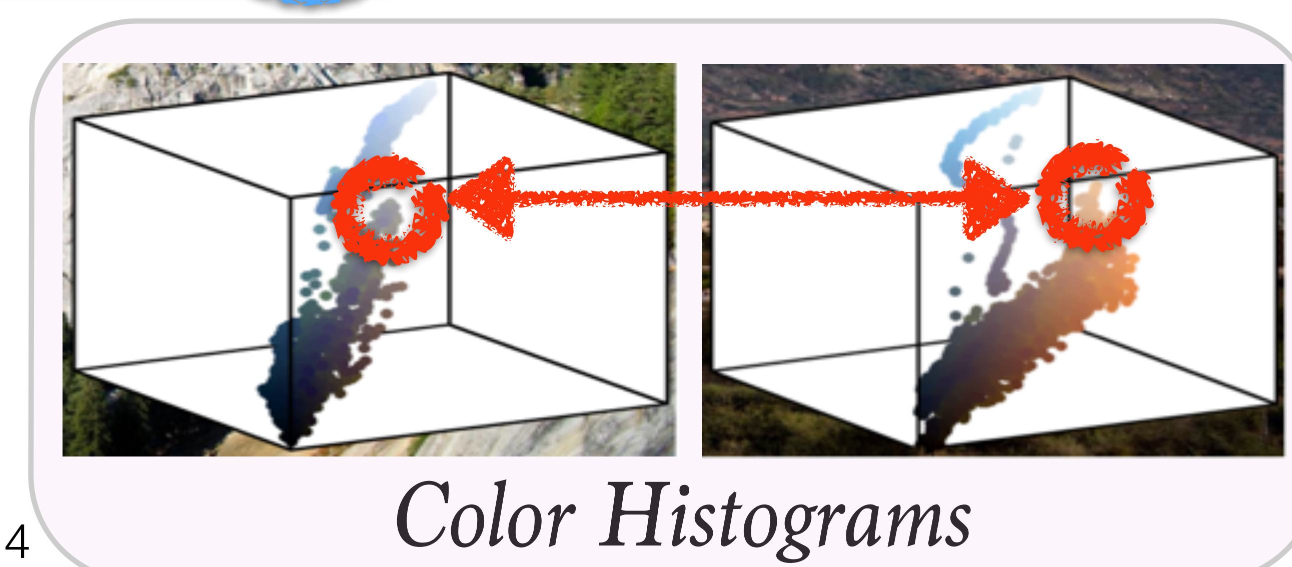
What is Optimal Transport?

A geometric toolbox to
compare probability measures
supported on a **metric space**.



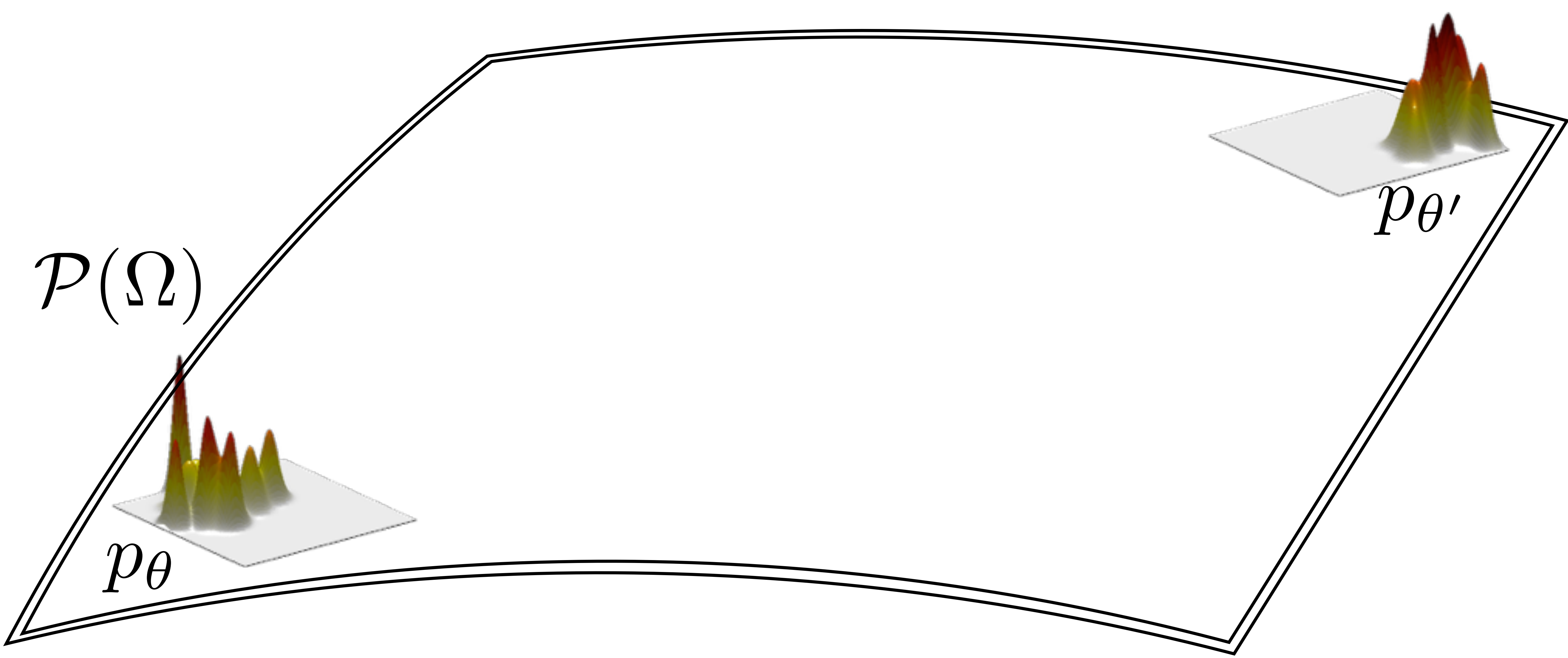






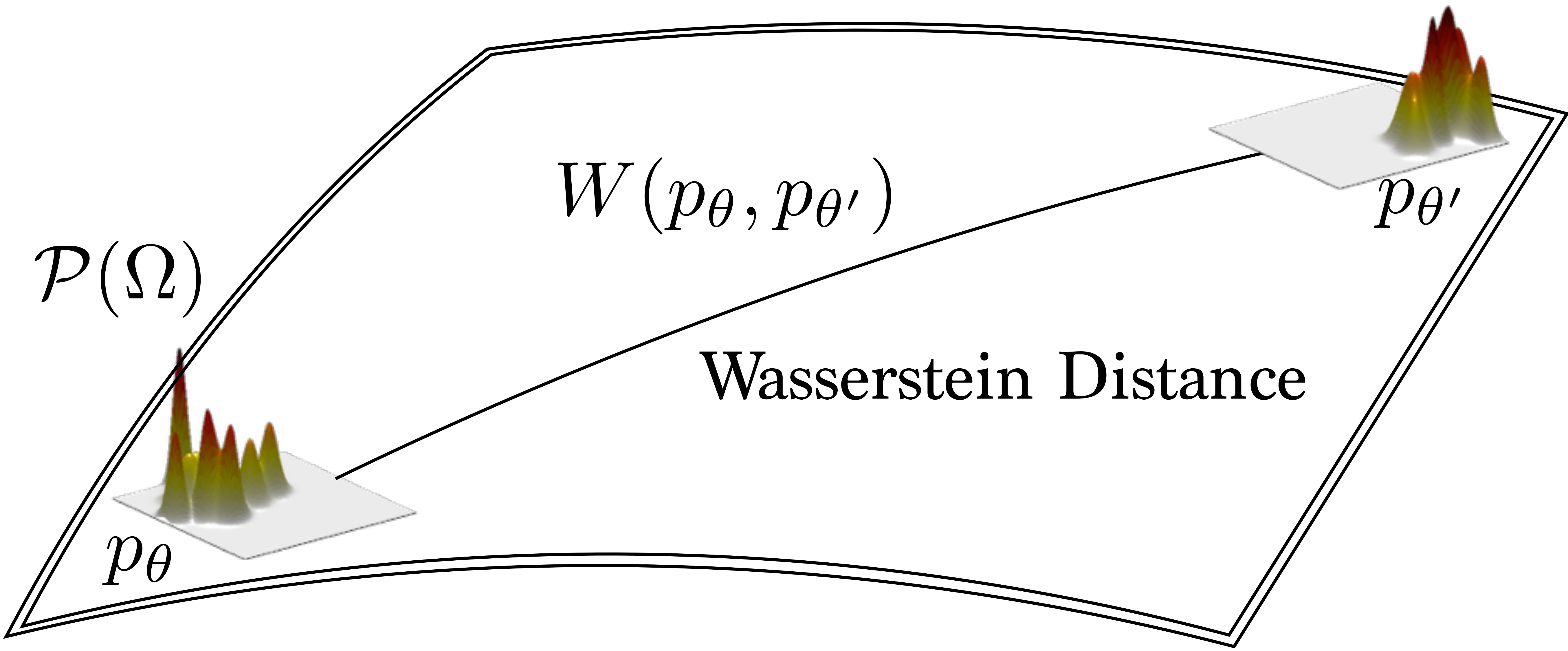
What is Optimal Transport?

A **geometric toolbox** to
compare probability measures
supported on a metric space.



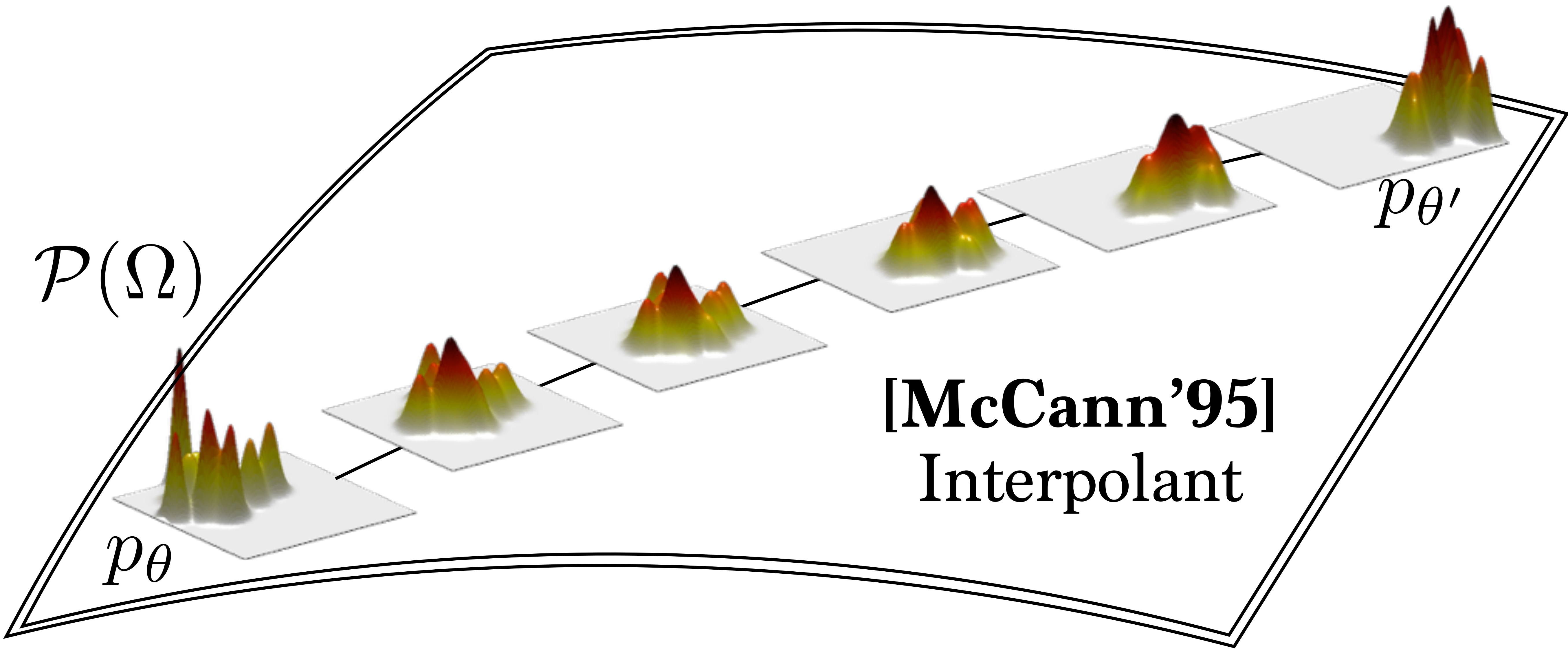
What is Optimal Transport?

A **geometric toolbox** to
compare probability measures
supported on a metric space.



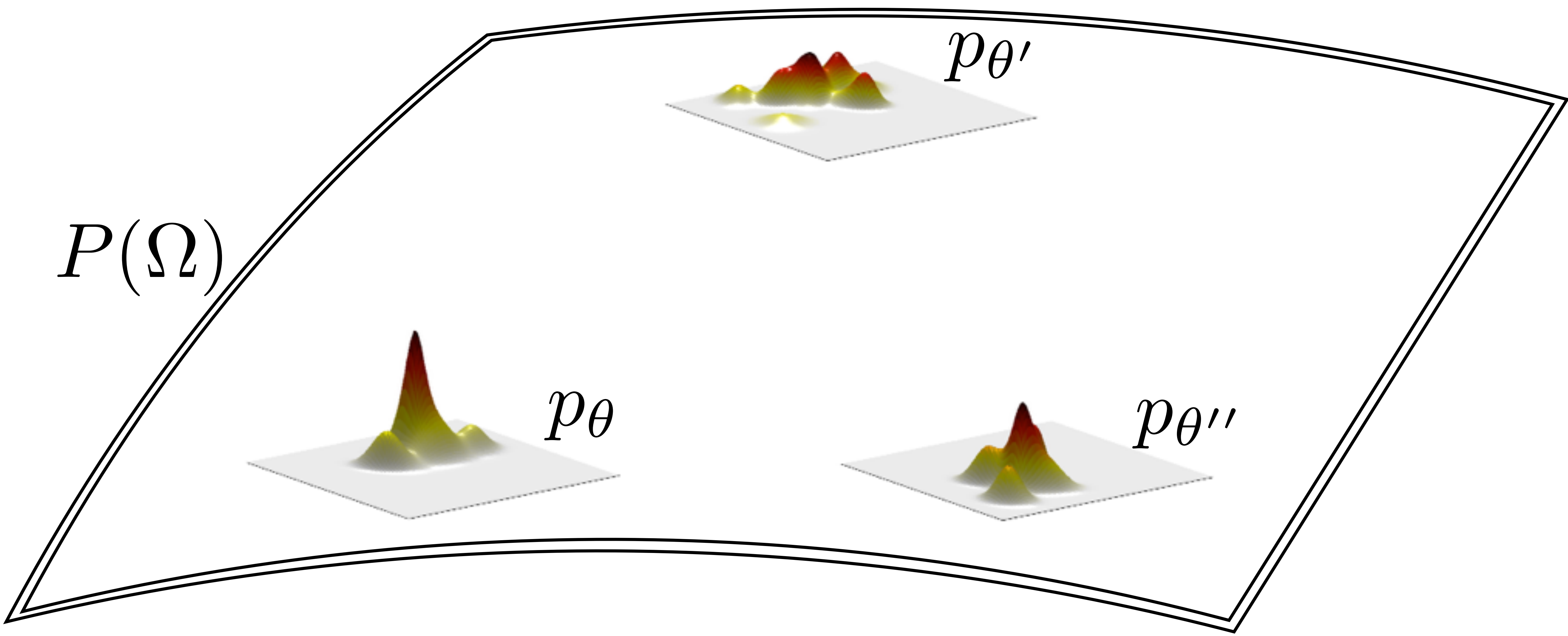
What is Optimal Transport?

A **geometric toolbox** to compare probability measures supported on a metric space.



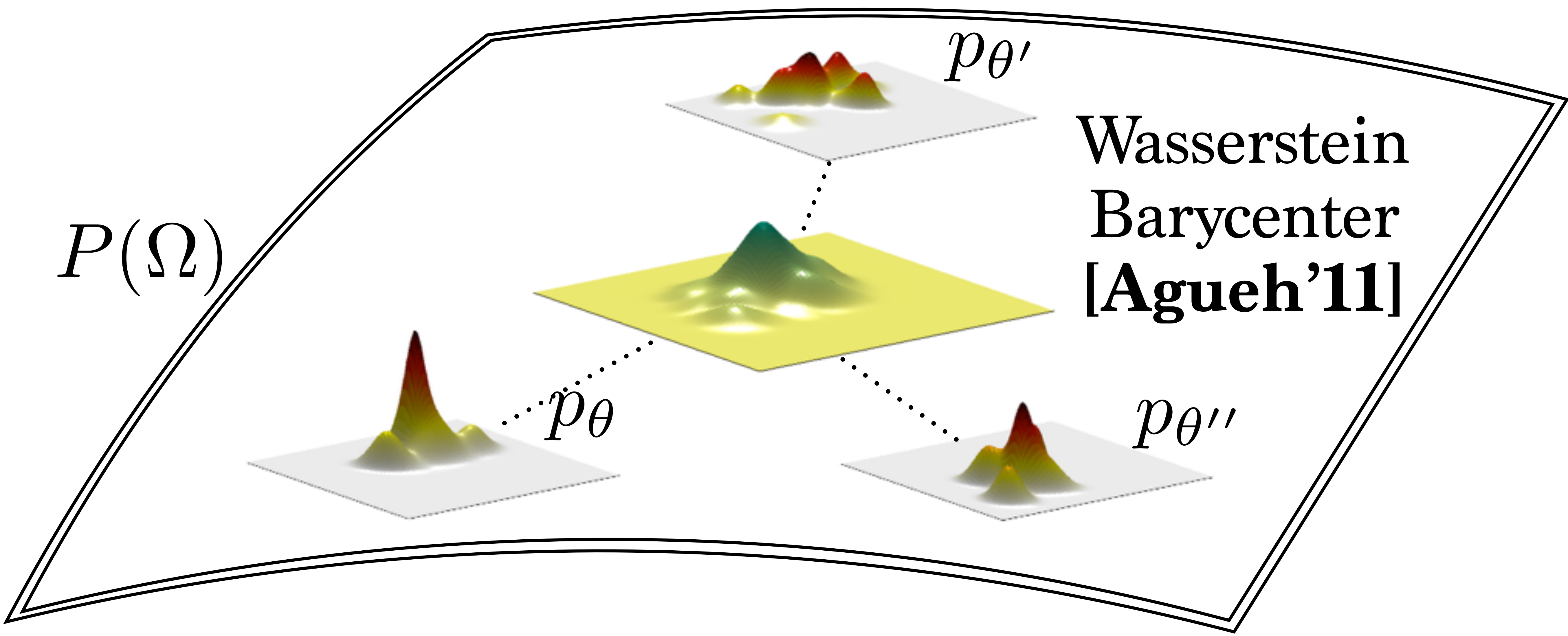
What is Optimal Transport?

A **geometric toolbox** to
compare probability measures
supported on a metric space.



What is Optimal Transport?

A **geometric toolbox** to
compare probability measures
supported on a metric space.



OT and data-analysis

- Key developments in (applied) maths ~'90s
  [McCann'95], [JKO'98], [Benamou'98], [Gangbo'98], [Ambrosio'06], [Villani'03/'09].

- Key developments in TCS / graphics since '00s
  [Rubner'98], [Indyk'03], [Naor'07], [Andoni'15].

◉ Small to *no-impact* in large-scale data analysis:

- ✦ **computationally heavy;**
- ✦ **Wasserstein distance is not differentiable**

OT and data-analysis

Today's talk: Entropy Regularized OT

- **Very fast** compared to usual approaches, GPGPU parallel.
- **Differentiable**, important if we want to use OT distances as **loss functions**.
- Can be **automatically differentiated**, simple iterative process, *DL*-toolboxes compatible.
- OT can become a building block in ML.

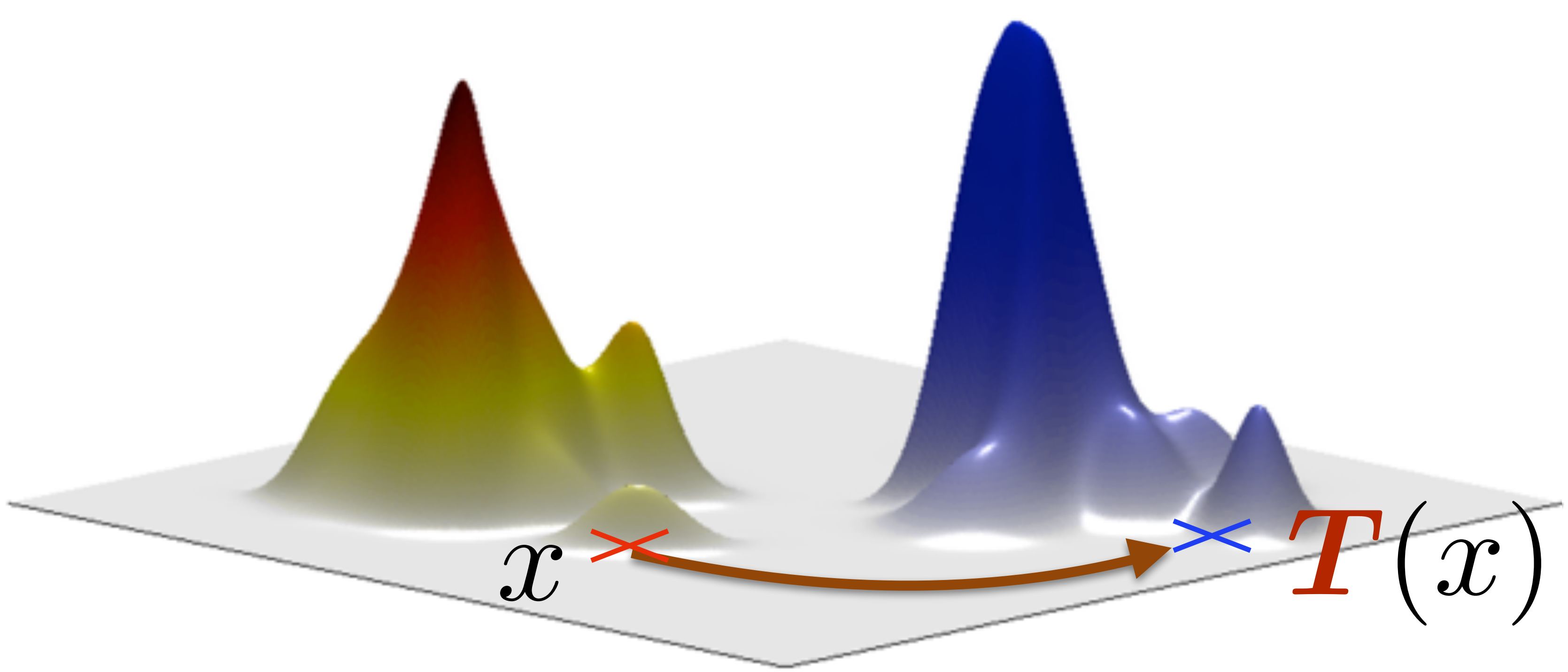
✦ Wasserstein distance is not differentiable

Background: OT Geometry

Consider (Ω, D), a metric probability space.
Let μ, ν be probability measures in $\mathcal{P}(\Omega)$.

- **[Monge'81]** problem: find a map $T : \Omega \to \Omega$

$$\inf_{T\#\mu=\nu} \int_\Omega D(x, T(x))\mu(dx)$$

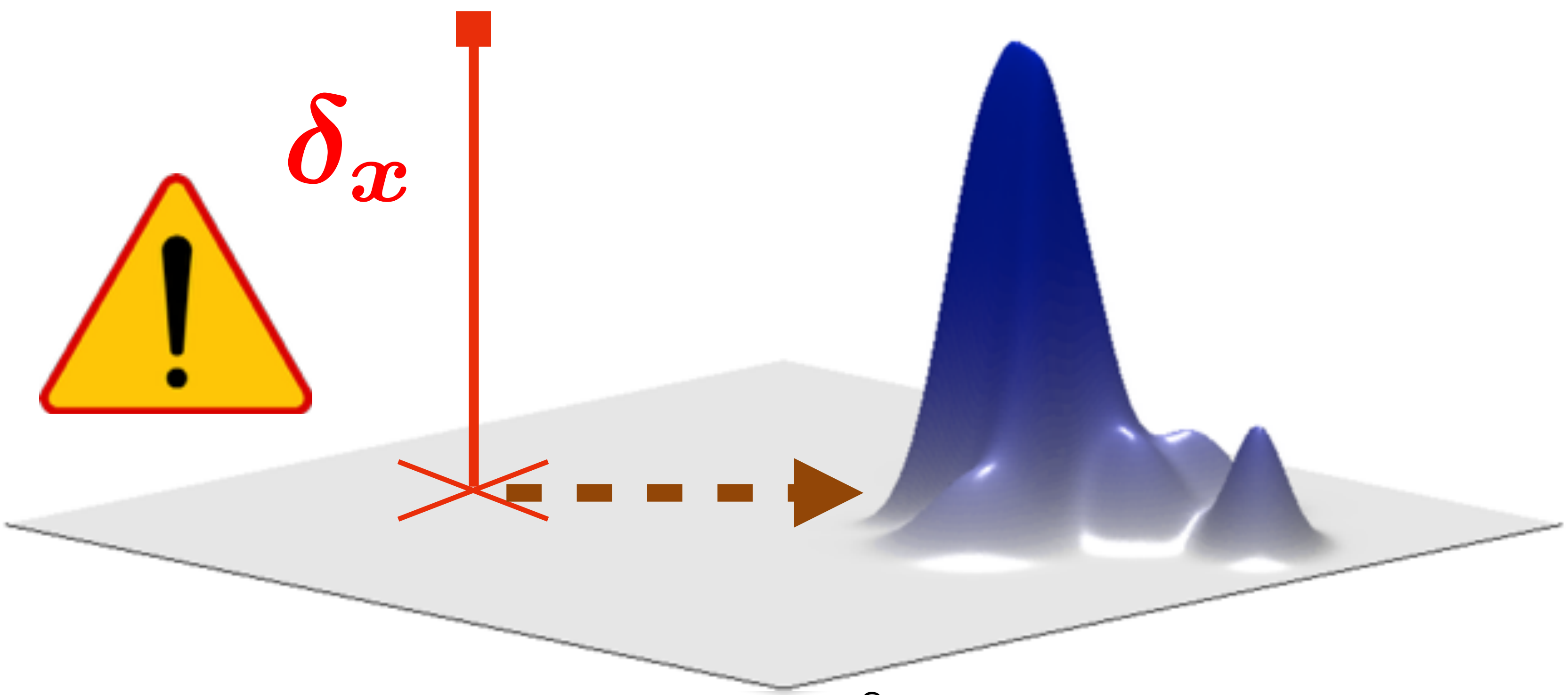


Background: OT Geometry

Consider (Ω, D), a metric probability space.
Let μ, ν be probability measures in $\mathcal{P}(\Omega)$.

- **[Monge'81]** problem: find a map $T : \Omega \to \Omega$

$$\inf_{T\#\mu=\nu} \int_\Omega D(x, T(x))\mu(dx)$$



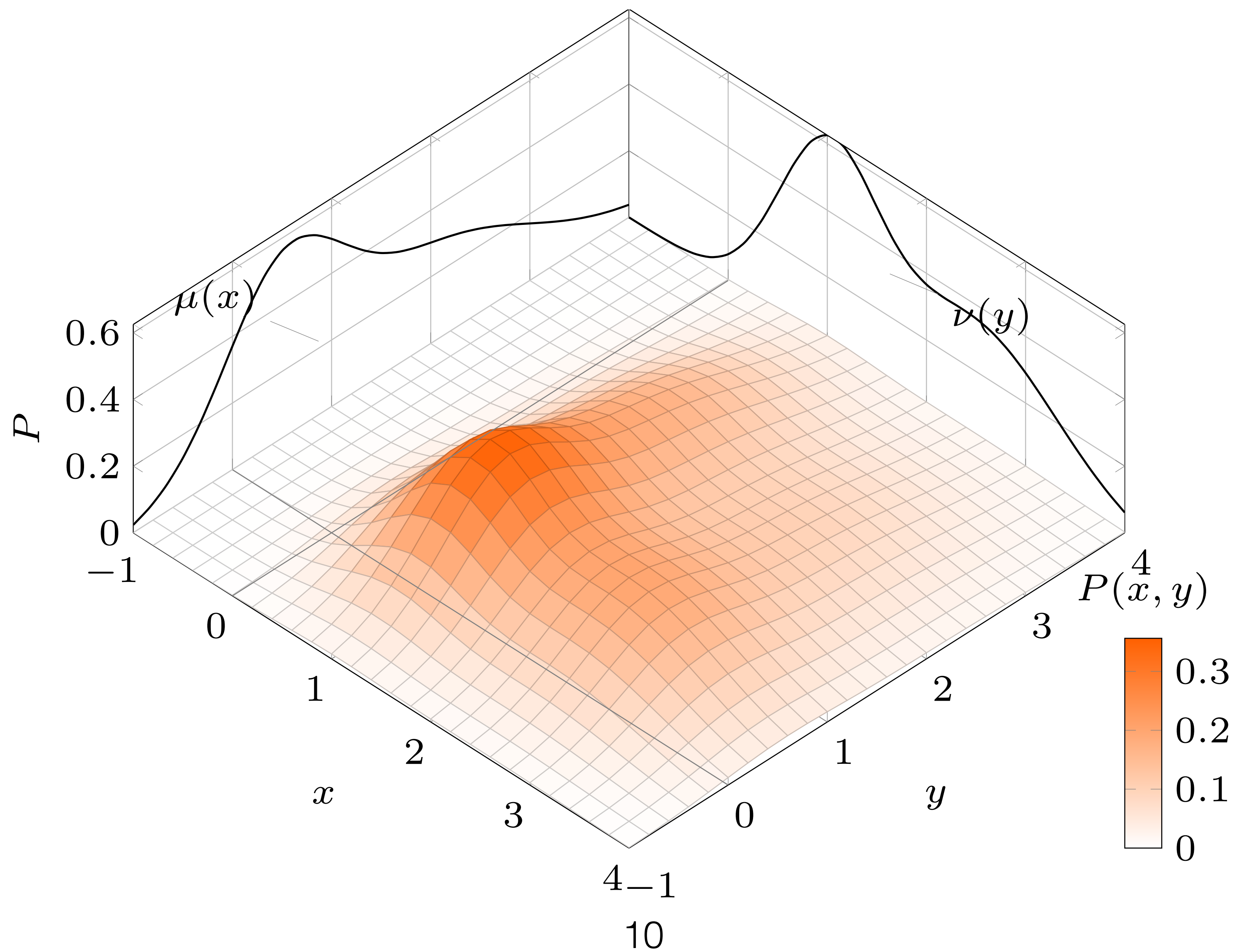
[Kantorovich'42] Relaxation

- Instead of maps $\boldsymbol{T} : \Omega \rightarrow \Omega$, consider probabilistic maps, i.e. **couplings** $\boldsymbol{P} \in \mathcal{P}(\Omega \times \Omega)$:

$$\Pi(\boldsymbol{\mu}, \boldsymbol{\nu}) \stackrel{\text{def}}{=} \{\boldsymbol{P} \in \mathcal{P}(\Omega \times \Omega) | \forall \boldsymbol{A}, \boldsymbol{B} \subset \Omega, \\ \boldsymbol{P}(\boldsymbol{A} \times \Omega) = \boldsymbol{\mu}(\boldsymbol{A}), \\ \boldsymbol{P}(\Omega \times \boldsymbol{B}) = \boldsymbol{\nu}(\boldsymbol{B})\}$$

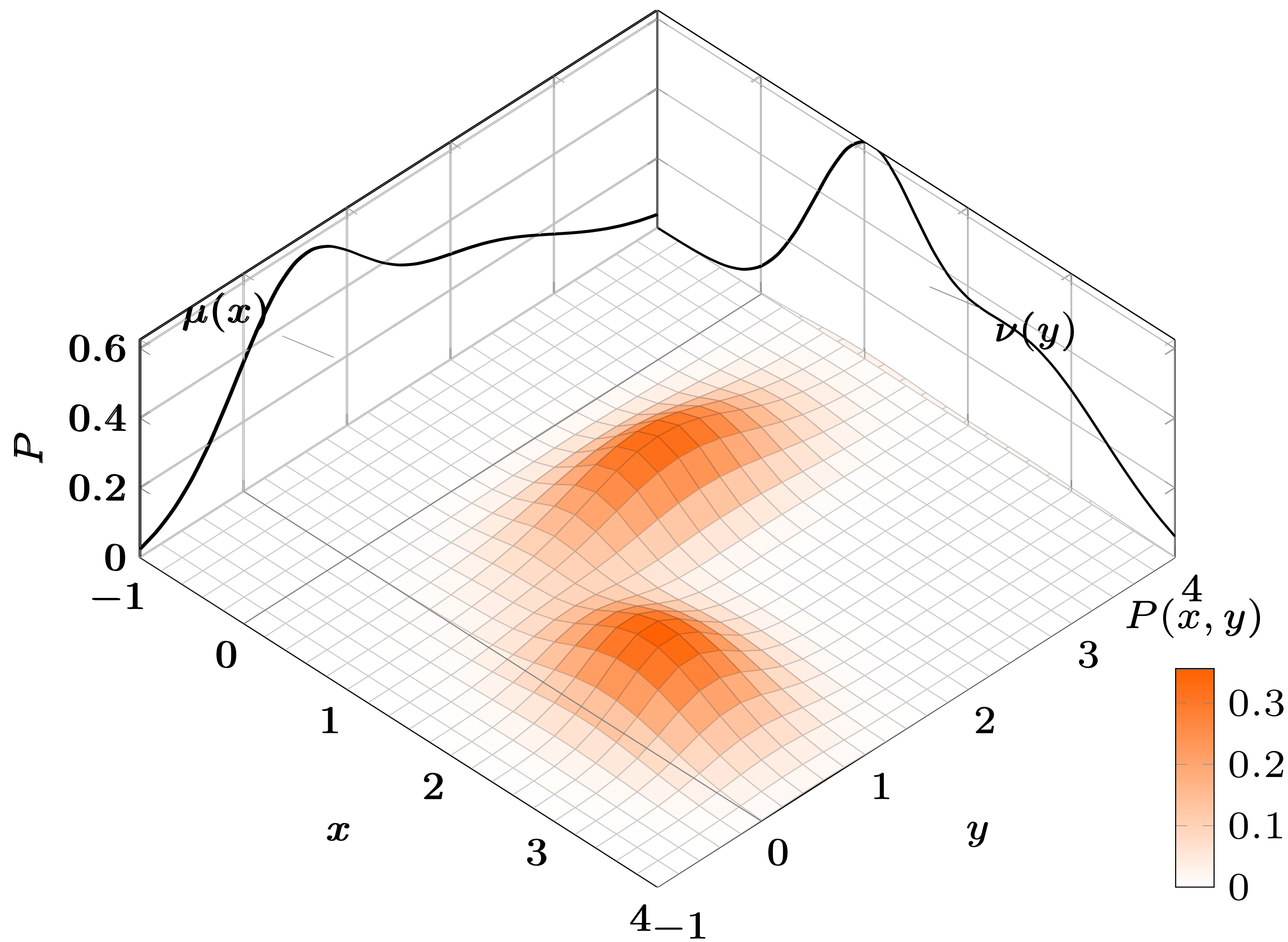
[Kantorovich'42] Relaxation

$$\Pi(\boldsymbol{\mu},\boldsymbol{\nu}) \stackrel{\text{def}}{=} \{\boldsymbol{P} \in \mathcal{P}(\Omega \times \Omega) \,|\, \forall \boldsymbol{A}, \boldsymbol{B} \subset \Omega, \\ \boldsymbol{P}(\boldsymbol{A} \times \Omega) = \boldsymbol{\mu}(\boldsymbol{A}), \boldsymbol{P}(\Omega \times \boldsymbol{B}) = \boldsymbol{\nu}(\boldsymbol{B})\}$$

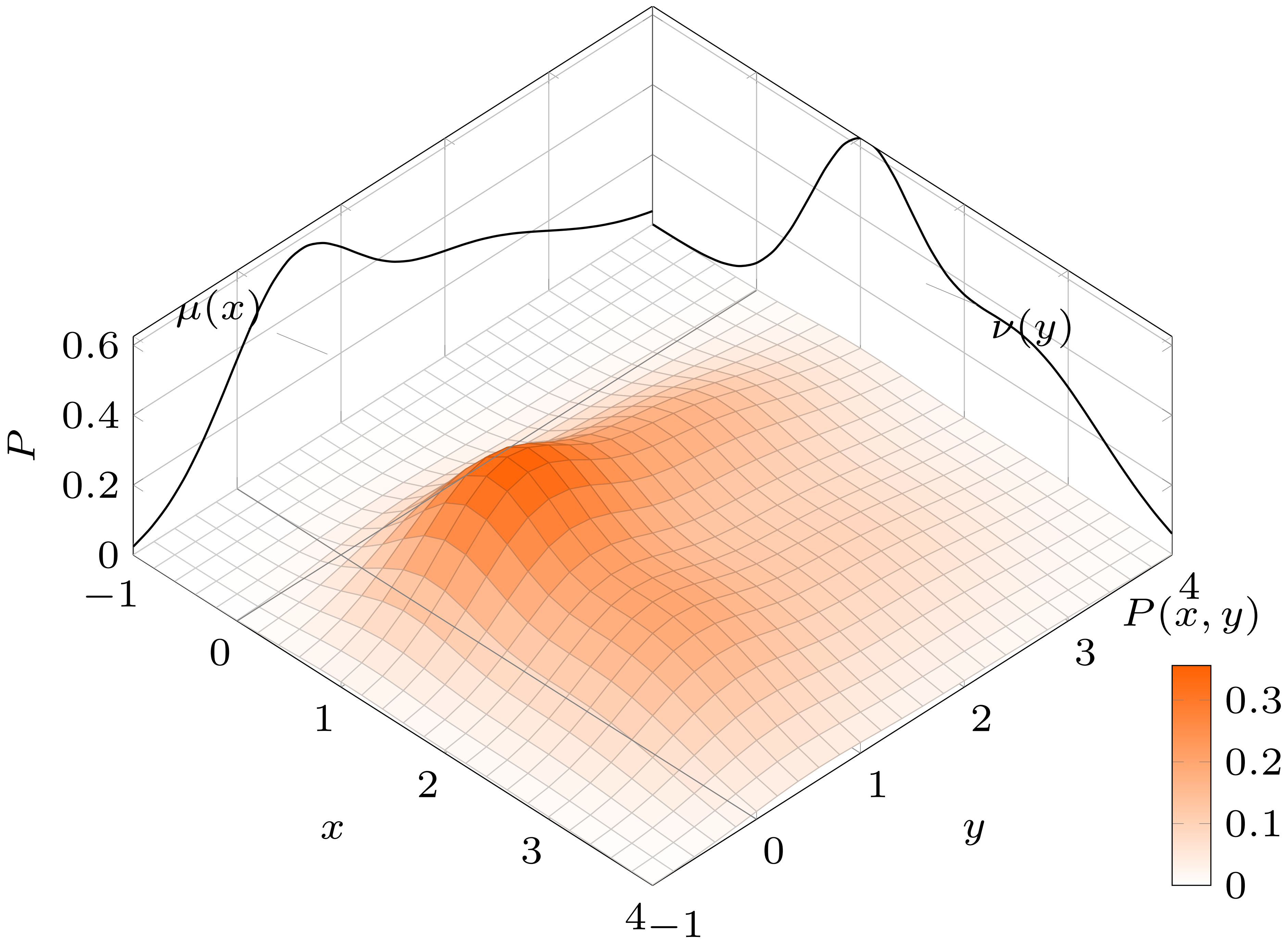


[Kantorovich'42] Relaxation

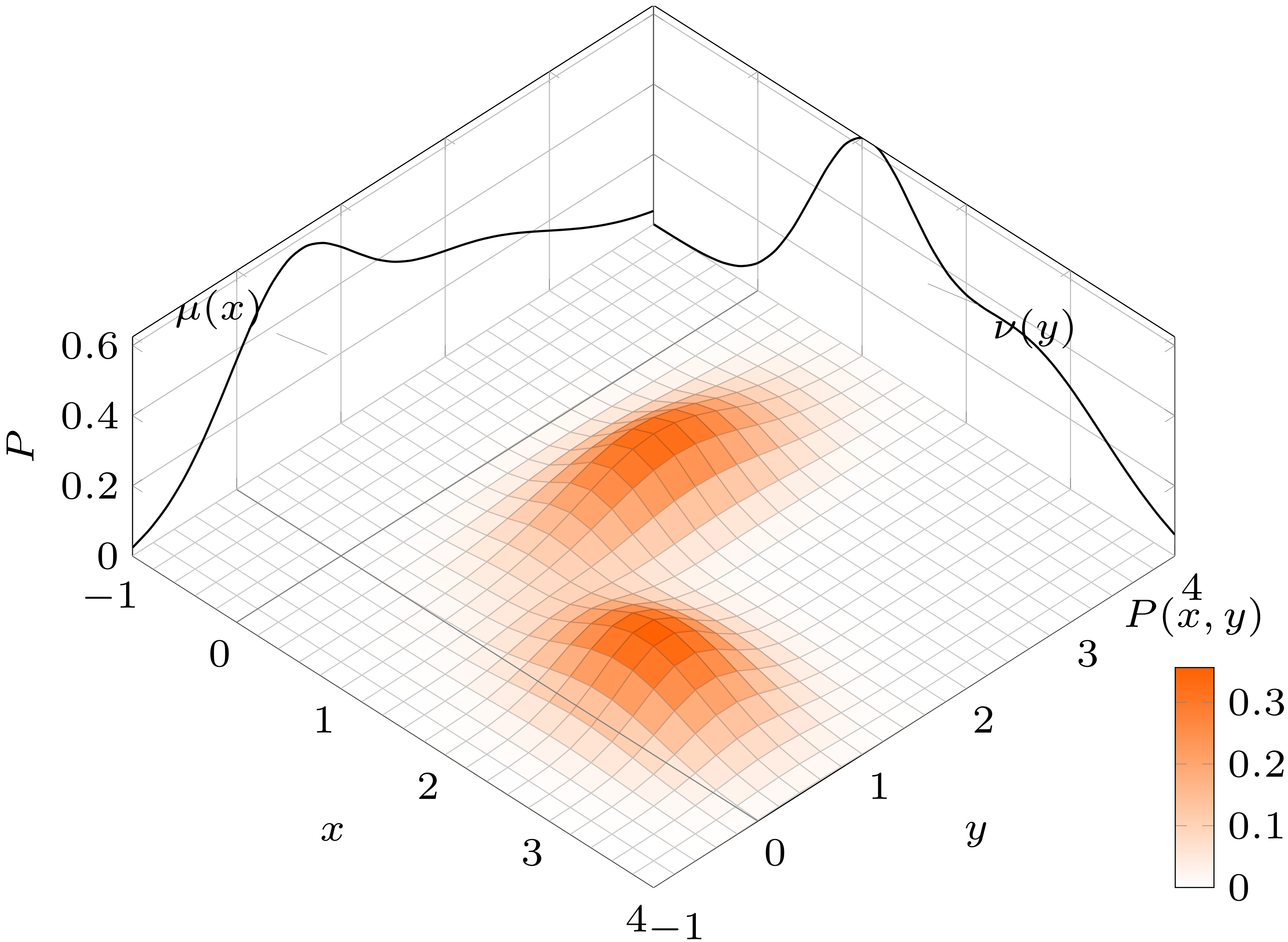
$$\Pi(\mu, \nu) \stackrel{\text{def}}{=} \{P \in \mathcal{P}(\Omega \times \Omega) | \forall A, B \subset \Omega,$$
$$P(A \times \Omega) = \mu(A), P(\Omega \times B) = \nu(B)\}$$



Couplings



Couplings

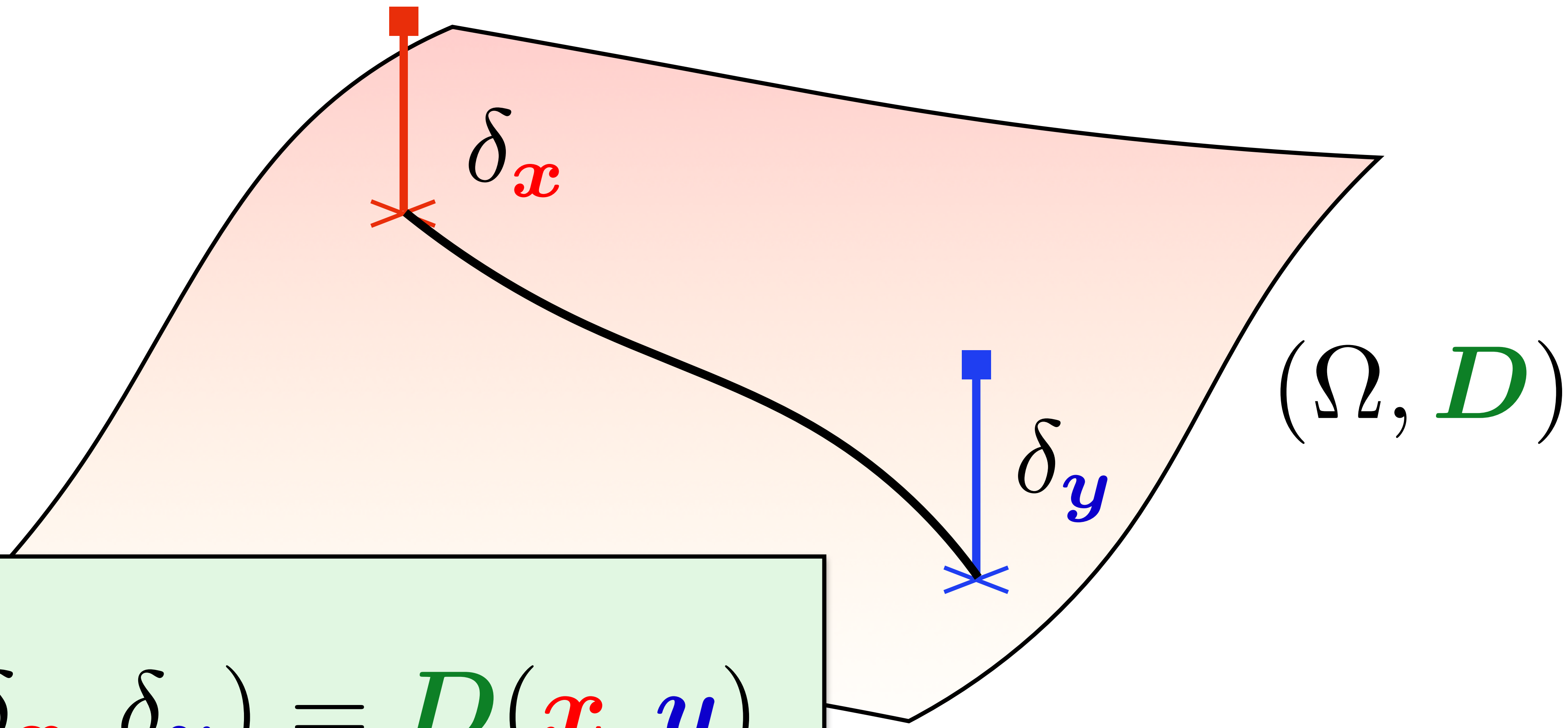


Wasserstein Distance

Def. For $p \geq 1$, the p-Wasserstein distance between $\boldsymbol{\mu}, \boldsymbol{\nu}$ in $\mathcal{P}(\Omega)$ is

$$W_p(\boldsymbol{\mu}, \boldsymbol{\nu}) \stackrel{\text{def}}{=} \left(\inf_{\boldsymbol{P} \in \Pi(\boldsymbol{\mu}, \boldsymbol{\nu})} \mathbb{E}_{\boldsymbol{P}}[D(X, Y)^p] \right)^{1/p}.$$

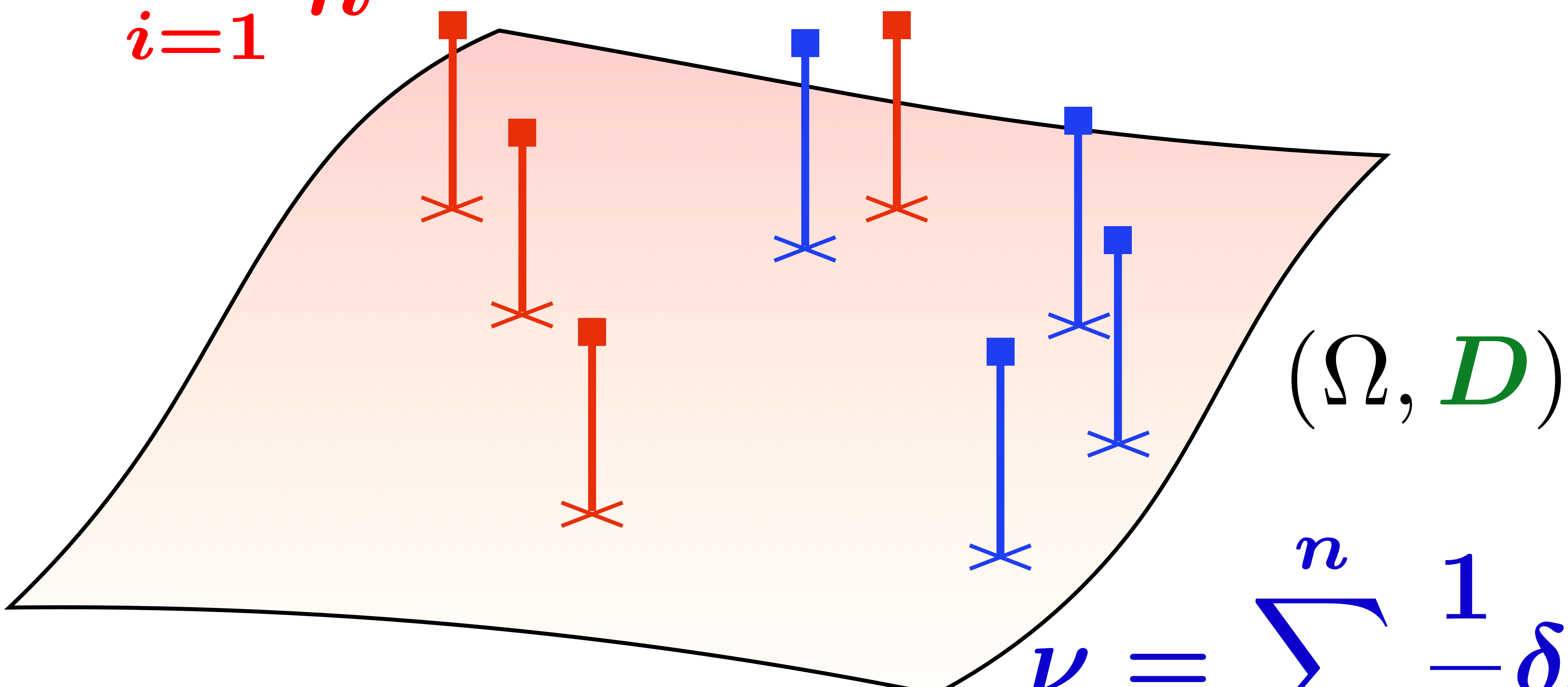
Wasserstein between 2 Diracs



$$W_p^p(\delta_{\boldsymbol{x}}, \delta_{\boldsymbol{y}}) = \boldsymbol{D}(\boldsymbol{x}, \boldsymbol{y})$$

Wasserstein on Uniform Measures

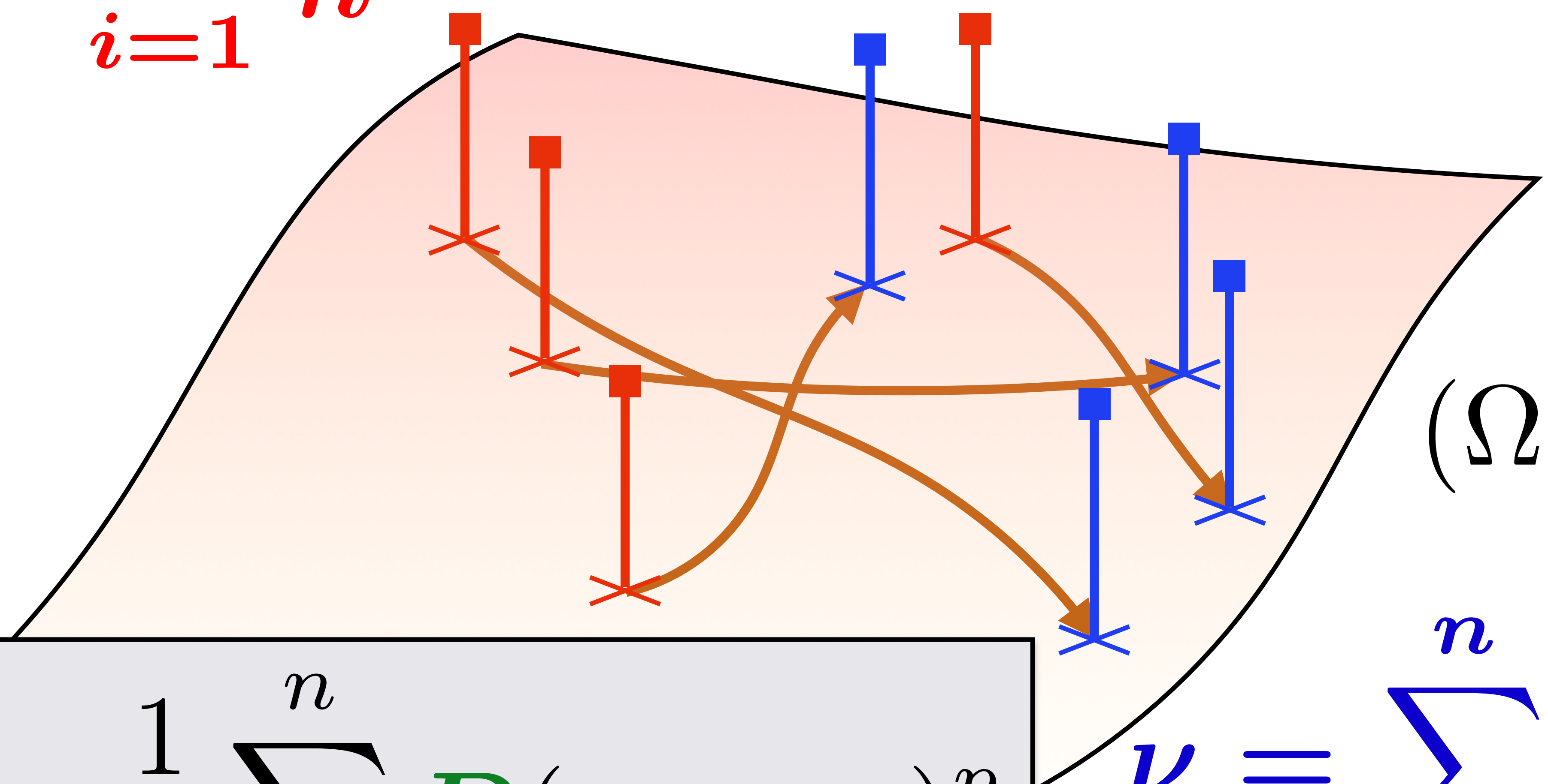
$$\mu = \sum_{i=1}^{n} \frac{1}{n} \delta_{x_i}$$



$$\nu = \sum_{j=1}^{n} \frac{1}{n} \delta_{y_j}$$

Wasserstein on Uniform Measures

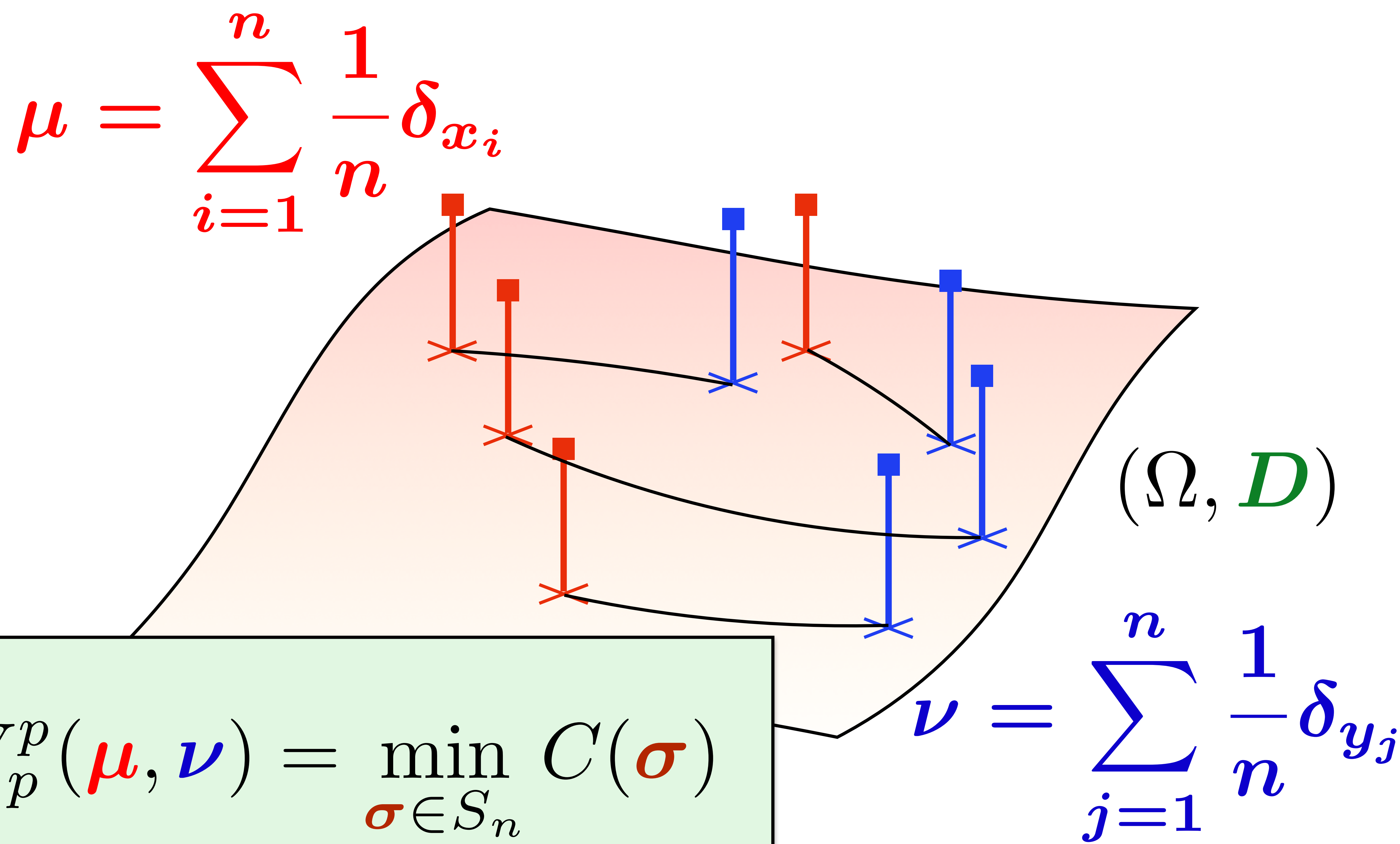
$$\mu = \sum_{i=1}^{n} \frac{1}{n} \delta_{x_i}$$



$$\nu = \sum_{j=1}^{n} \frac{1}{n} \delta_{y_j}$$

$$C(\sigma) = \frac{1}{n} \sum_{i=1}^{n} D(x_i, y_{\sigma_i})^p$$

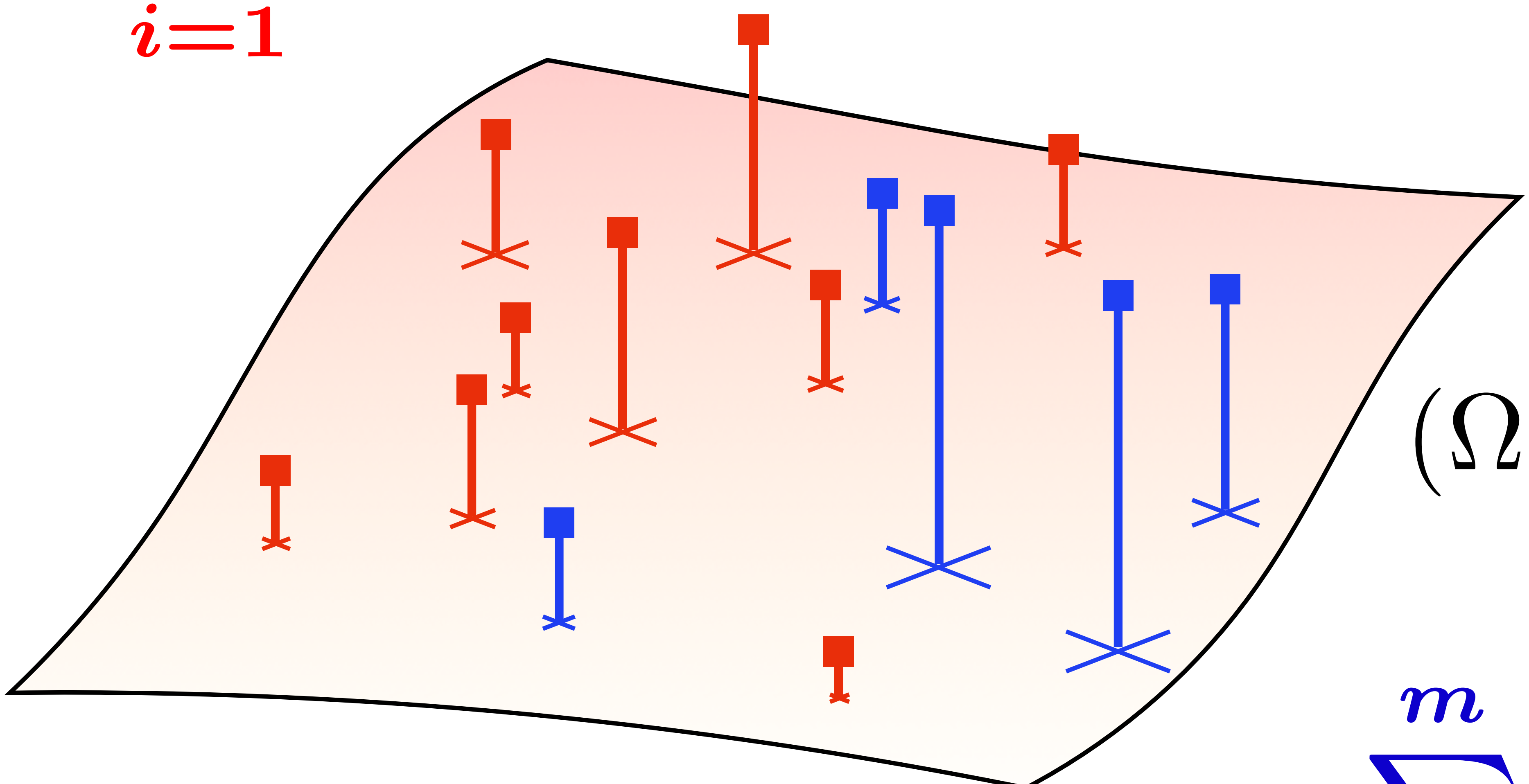
Optimal Assignment ⊂ Wasserstein



$$W_p^p(\mu, \nu) = \min_{\sigma \in S_n} C(\sigma)$$

Wasserstein on Empirical Measures

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



(Ω, D)

$$\nu = \sum_{j=1}^{m} b_j \delta_{y_j}$$

Wasserstein on Empirical Measures

Consider $\boldsymbol{\mu} = \sum_{i=1}^{n} \boldsymbol{a}_i \boldsymbol{\delta}_{\boldsymbol{x}_i}$ and $\boldsymbol{\nu} = \sum_{j=1}^{m} \boldsymbol{b}_j \boldsymbol{\delta}_{\boldsymbol{y}_j}$.

$$M_{\mathbf{XY}} \stackrel{\text{def}}{=} [D(\boldsymbol{x}_i, \boldsymbol{y}_j)^p]_{ij}$$

$$U(\boldsymbol{a}, \boldsymbol{b}) \stackrel{\text{def}}{=} \{\boldsymbol{P} \in \mathbb{R}_+^{n \times m} \,|\, \boldsymbol{P}\mathbf{1}_m = \boldsymbol{a}, \boldsymbol{P}^T\mathbf{1}_n = \boldsymbol{b}\}$$

$$\begin{array}{c} \\ \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_n \end{array}
\begin{array}{c} \begin{array}{ccc} \boldsymbol{y}_1 & \dots & \boldsymbol{y}_m \end{array} \\ \left[\begin{array}{ccc} \cdot & \cdot & \cdot \\ \cdot & D(\boldsymbol{x}_i, \boldsymbol{y}_j)^p & \cdot \\ \cdot & \cdot & \cdot \end{array}\right] \end{array}
\qquad
\begin{array}{c} \\ \boldsymbol{a}_1 \\ \vdots \\ \boldsymbol{a}_n \end{array}
\begin{array}{c} \begin{array}{ccc} \boldsymbol{b}_1 & \dots & \boldsymbol{b}_m \end{array} \\ \left[\begin{array}{ccc} \cdots & \cdots & \cdots \\ \cdots & \boldsymbol{P}\mathbf{1}_m = \boldsymbol{a} & \cdots \\ \cdots & \cdots & \cdots \end{array}\right] \end{array}$$

Wasserstein on Empirical Measures

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

$$M_{XY} \stackrel{\text{def}}{=} [D(x_i, y_j)^p]_{ij}$$

$$U(a,b) \stackrel{\text{def}}{=} \{P \in \mathbb{R}_+^{n \times m} \mid P\mathbf{1}_m = a, P^T\mathbf{1}_n = b\}$$

$$\begin{array}{c} \\ x_1 \\ \vdots \\ x_n \end{array} \begin{array}{c} \begin{array}{ccc} y_1 & \cdots & y_m \end{array} \\ \left[\begin{array}{ccc} \cdot & \cdot & \cdot \\ \cdot & D(x_i, y_j)^p & \cdot \\ \cdot & \cdot & \cdot \end{array}\right] \end{array} \qquad \begin{array}{c} \\ a_1 \\ \vdots \\ a_n \end{array} \begin{array}{c} \begin{array}{ccc} b_1 & \cdots & b_m \end{array} \\ \left[\begin{array}{ccc} \vdots & \vdots & \vdots \\ \vdots & P^T\mathbf{1}_n = b & \vdots \\ \vdots & \vdots & \vdots \end{array}\right] \end{array}$$

Wasserstein on Empirical Measures

Consider $\boldsymbol{\mu} = \sum_{i=1}^{n} \boldsymbol{a}_i \boldsymbol{\delta}_{\boldsymbol{x}_i}$ and $\boldsymbol{\nu} = \sum_{j=1}^{m} \boldsymbol{b}_j \boldsymbol{\delta}_{\boldsymbol{y}_j}$.

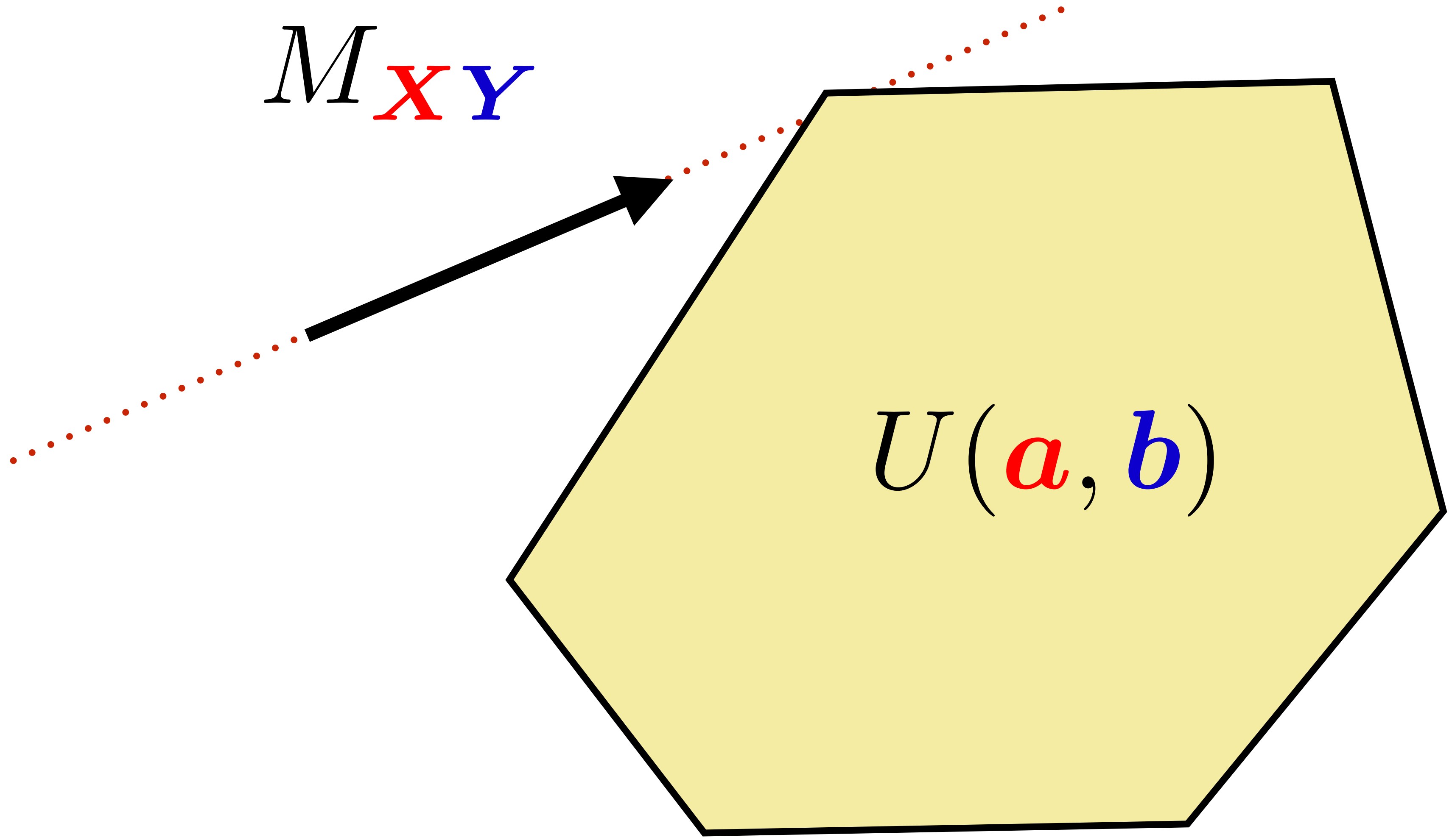
$$M_{\mathbf{XY}} \stackrel{\text{def}}{=} [D(\boldsymbol{x}_i, \boldsymbol{y}_j)^p]_{ij}$$

$$U(\boldsymbol{a}, \boldsymbol{b}) \stackrel{\text{def}}{=} \{\boldsymbol{P} \in \mathbb{R}_+^{n \times m} \,|\, \boldsymbol{P}\mathbf{1}_m = \boldsymbol{a}, \boldsymbol{P}^T \mathbf{1}_n = \boldsymbol{b}\}$$

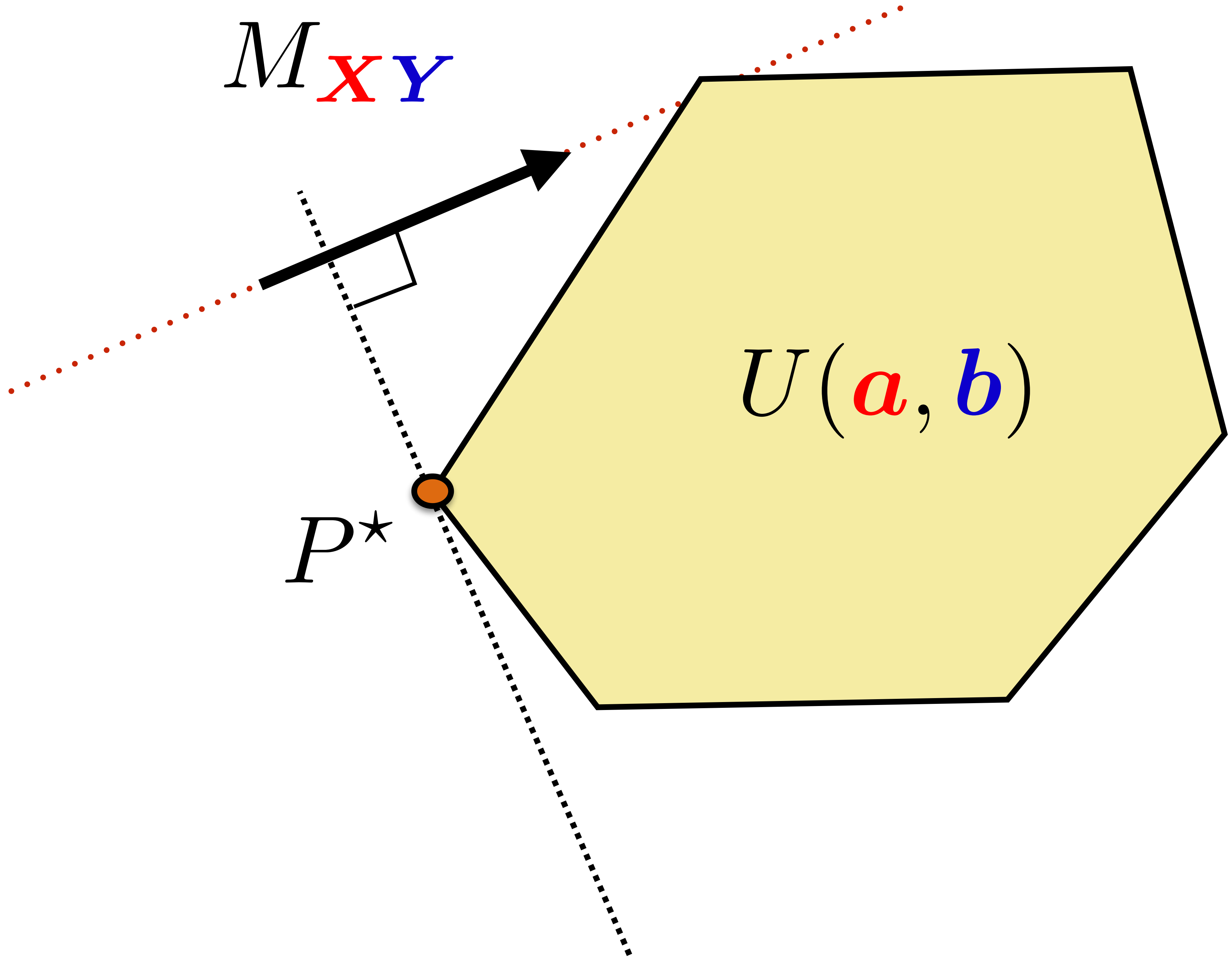
Def. Optimal Transport Problem

$$W_p^p(\boldsymbol{\mu}, \boldsymbol{\nu}) = \min_{\boldsymbol{P} \in U(\boldsymbol{a}, \boldsymbol{b})} \langle \boldsymbol{P}, M_{\mathbf{XY}} \rangle$$

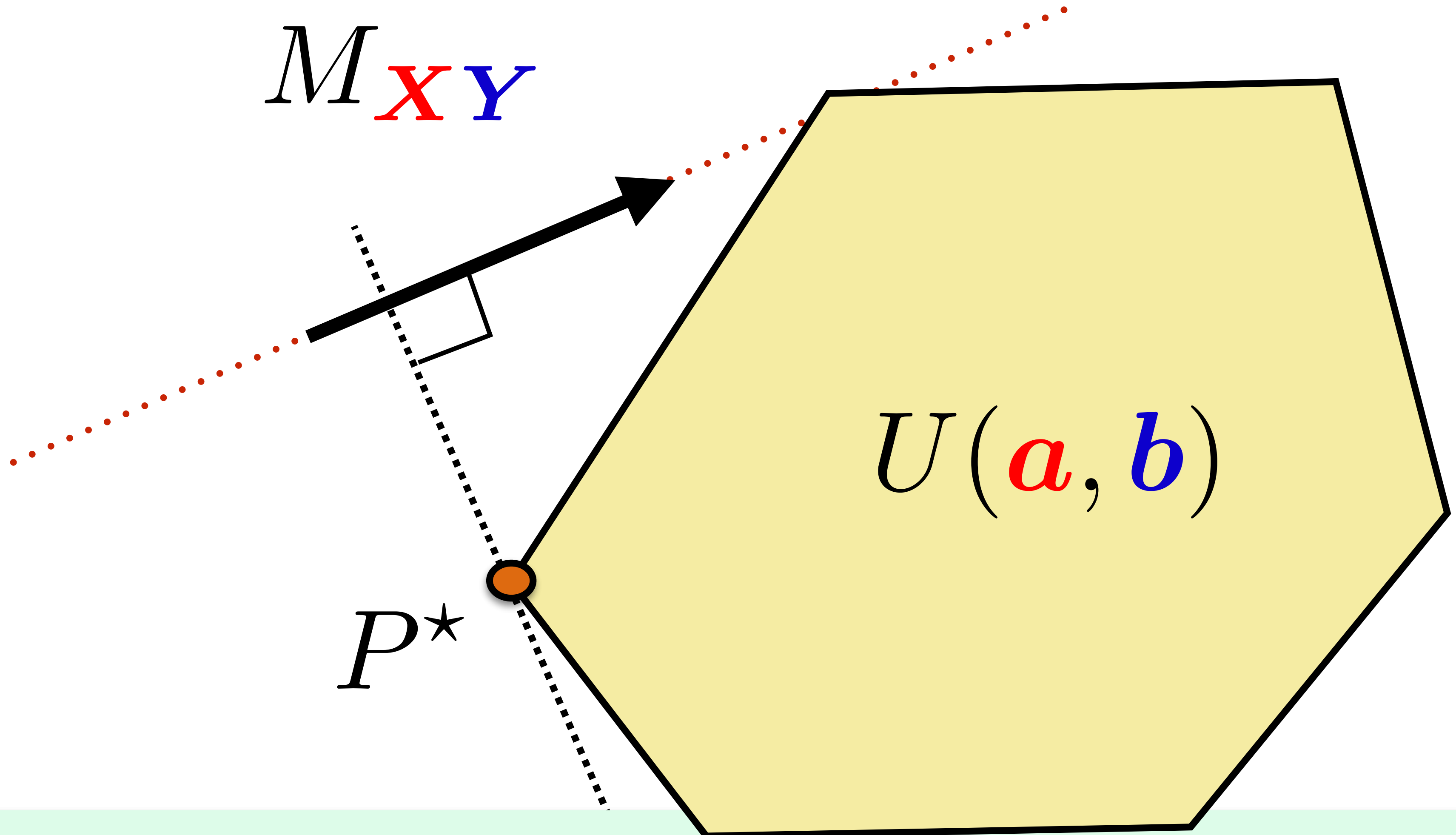
Discrete OT Problem



Discrete OT Problem



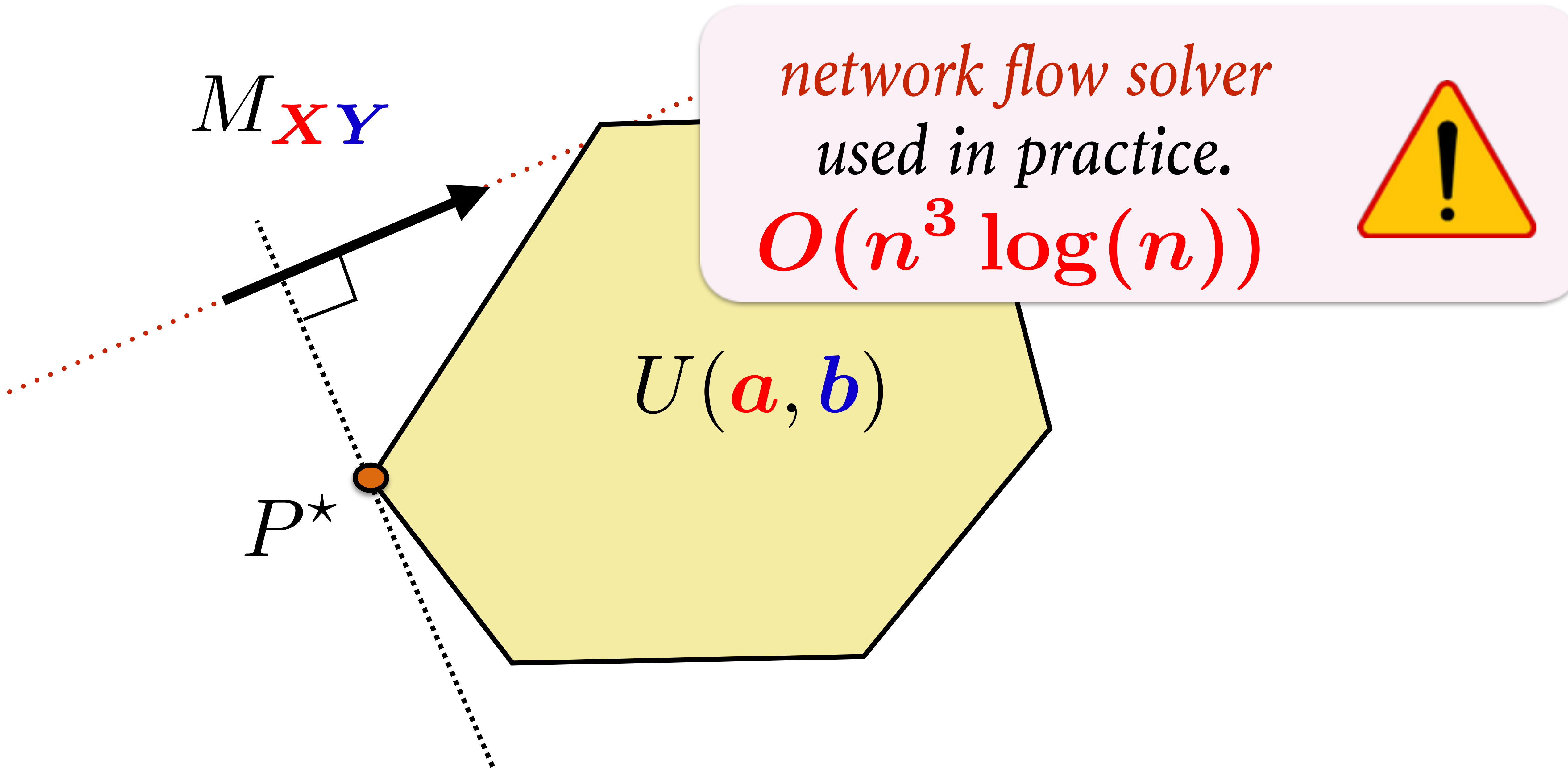
Discrete OT Problem



Def. Dual OT problem

$$W_p^p(\mu, \nu) = \max_{\substack{\alpha \in \mathbb{R}^n, \beta \in \mathbb{R}^m \\ \alpha_i + \beta_j \leq D(x_i, y_j)^p}} \alpha^T a + \beta^T b$$

Discrete OT Problem



Note: flow/PDE formulations **[Beckman'61]/[Benamou'98]** can be used for *p=1/p=2* for a sparse-graph metric/Euclidean metric.

Discrete OT Problem

```c
/*
    emd.c

    Last update: 3/14/98

    An implementation of the Earth Movers Distance.
    Based of the solution for the Transportation problem as described in
    "Introduction to Mathematical Programming" by F. S. Hillier and
    G. J. Lieberman, McGraw-Hill, 1990.

    Copyright (C) 1998 Yossi Rubner
    Computer Science Department, Stanford University
    E-Mail: rubner@cs.stanford.edu   URL: http://vision.stanford.edu/~rubner
*/

/*#include <stdio.h>
#include <stdlib.h>*/
#include <math.h>

#include "emd.h"

#define DEBUG_LEVEL 0
/*
 DEBUG_LEVEL:
   0 = NO MESSAGES
   1 = PRINT THE NUMBER OF ITERATIONS AND THE FINAL RESULT
   2 = PRINT THE RESULT AFTER EVERY ITERATION
   3 = PRINT ALSO THE FLOW AFTER EVERY ITERATION
   4 = PRINT A LOT OF INFORMATION (PROBABLY USEFUL ONLY FOR THE AUTHOR)
*/


#define MAX_SIG_SIZE1  (MAX_SIG_SIZE+1)  /* FOR THE POSIBLE DUMMY FEATURE */

/* NEW TYPES DEFINITION */

/* node1_t IS USED FOR SINGLE-LINKED LISTS */
typedef struct node1_t {
  int i;
  double val;
  struct node1_t *Next;
} node1_t;

/* node1_t IS USED FOR DOUBLE-LINKED LISTS */
typedef struct node2_t {
  int i, j;
  double val;
  struct node2_t *NextC;               /* NEXT COLUMN */
  struct node2_t *NextR;               /* NEXT ROW */
} node2_t;



/* GLOBAL VARIABLE DECLARATION */
static int _n1, _n2;                          /* SIGNATURES SIZES */
static float _C[MAX_SIG_SIZE1][MAX_SIG_SIZE1];/* THE COST MATRIX */
static node2_t _X[MAX_SIG_SIZE1*2];            /* THE BASIC VARIABLES VECTOR */
```

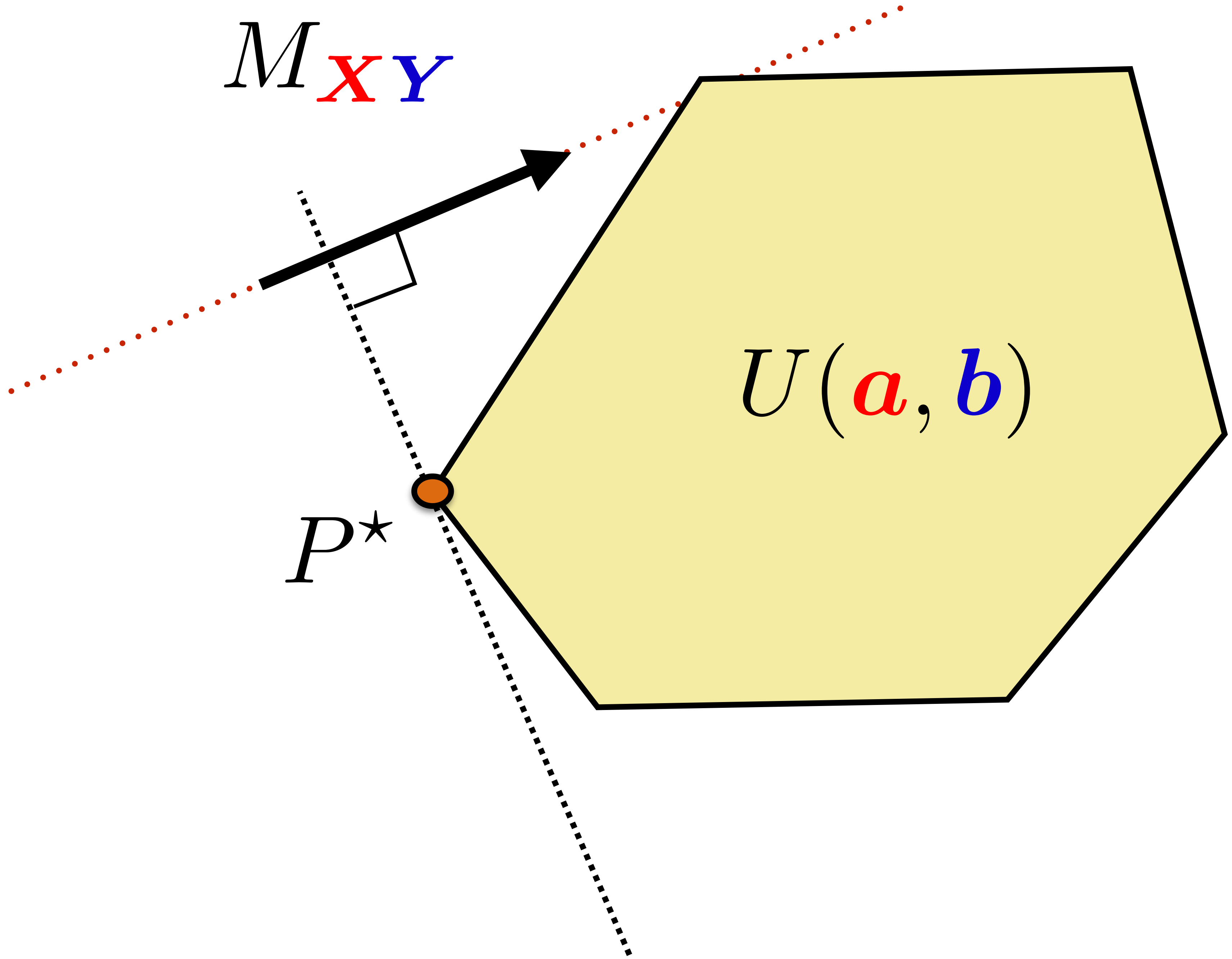
Discrete OT Problem

```
/*
    emd.c

    Last update: 3/14/98

    An implementation of the Earth Movers Distance.
    Based of the solution for the Transportation problem as described in
    "Introduction to Mathematical Programming" by F. S. Hillier and
    G. J. Lieberman, McGraw-Hill, 1990.

    Copyright (C) 1998 Yossi Rubner
    Computer Science Department, Stanford University
    E-Mail: rubner@cs.stanford.edu   URL: http://vision.stanford.edu/~rubner
*/

/*#include <stdio.h>
#include <stdlib.h>*/
#include <math.h>

#include "emd.h"

#define DEBUG_LEVEL 0
/*
 DEBUG_LEVEL:
   0 = NO MESSAGES
   1 = PRINT THE NUMBER OF ITERATIONS AND THE FINAL RESULT
   2 = PRINT THE RESULT AFTER EVERY ITERATION
   3 = PRINT ALSO THE FLOW AFTER EVERY ITERATION
   4 = PRINT A LOT OF INFORMATION (PROBABLY USEFUL ONLY FOR THE AUTHOR)
*/


#define MAX_SIG_SIZE1  (MAX_SIG_SIZE+1)  /* FOR THE POSIBLE DUMMY FEATURE */

/* NEW TYPES DEFINITION */

/* node1_t IS USED FOR SINGLE-LINKED LISTS */
typedef struct node1_t {
  int i;
  double val;
  struct node1_t *Next;
} node1_t;

/* node1_t IS USED FOR DOUBLE-LINKED LISTS */
typedef struct node2_t {
  int i, j;
  double val;
  struct node2_t *NextC;              /* NEXT COLUMN */
  struct node2_t *NextR;              /* NEXT ROW */
} node2_t;



/* GLOBAL VARIABLE DECLARATION */
static int _n1, _n2;                          /* SIGNATURES SIZES */
static float _C[MAX_SIG_SIZE1][MAX_SIG_SIZE1];/* THE COST MATRIX */
static node2_t _X[MAX_SIG_SIZE1*2];            /* THE BASIC VARIABLES VECTOR */
```

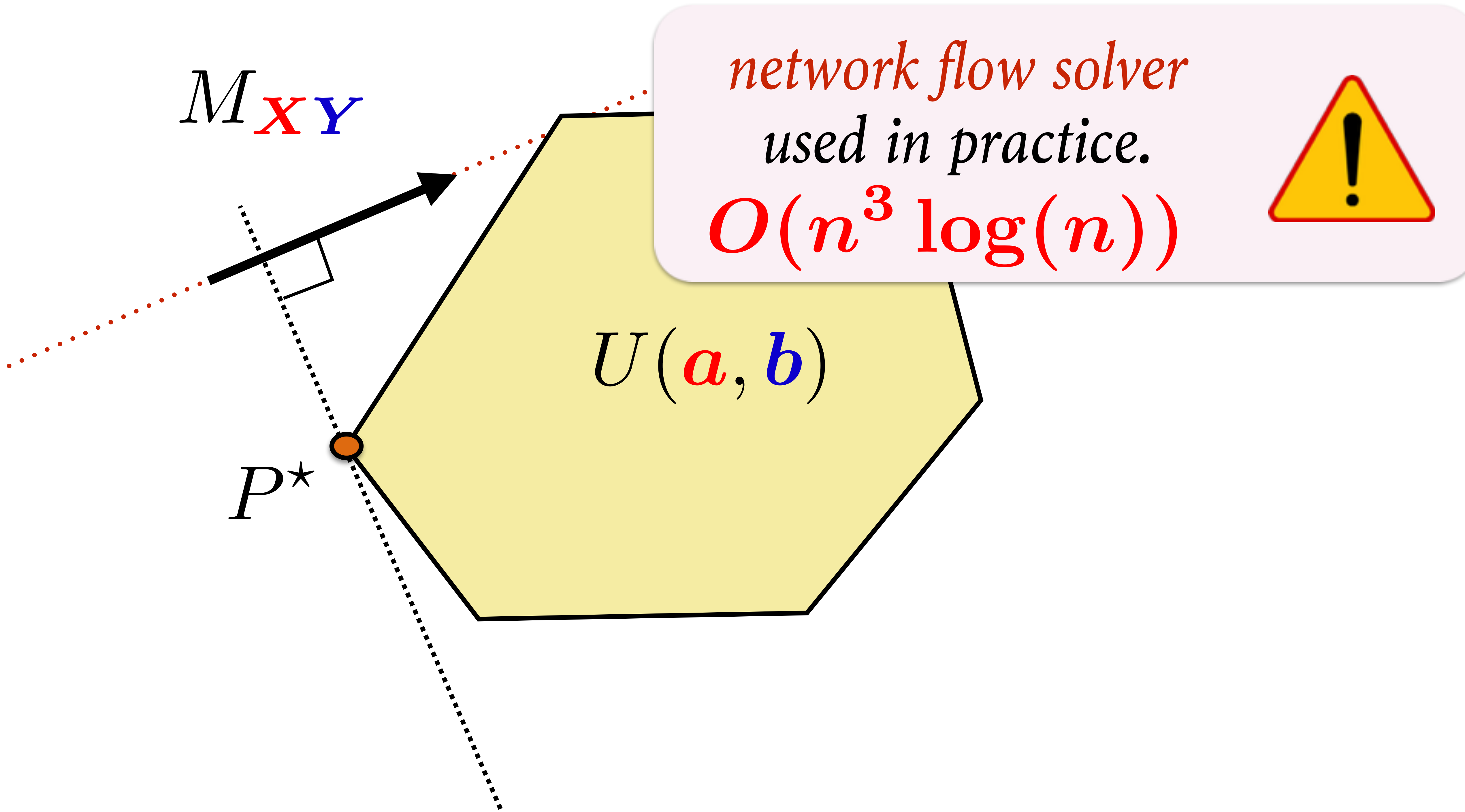
Discrete OT Problem

```c
/*
    emd.c

    Last update: 3/14/98

    An implementation of the Earth Movers Distance.
    Based of the solution for the Transportation problem as described in
    "Introduction to Mathematical Programming" by F. S. Hillier and
    G. J. Lieberman, McGraw-Hill, 1990.

    Copyright (C) 1998 Yossi Rubner
    Computer Science Department, Stanford University
    E-Mail: rubner@cs.stanford.edu   URL: http://vision.stanford.edu/~rubner
*/

/*#include <stdio.h>
#include <stdlib.h>*/
#include <math.h>

#include "emd.h"

#define DEBUG_LEVEL 0
/*
 DEBUG_LEVEL:
   0 = NO MESSAGES
   1 = PRINT THE NUMBER OF ITERATIONS AND THE FINAL RESULT
   2 = PRINT THE RESULT AFTER EVERY ITERATION
   3 = PRINT ALSO THE FLOW AFTER EVERY ITERATION
   4 = PRINT A LOT OF INFORMATION (PROBABLY USEFUL ONLY FOR THE AUTHOR)
*/


#define MAX_SIG_SIZE1  (MAX_SIG_SIZE+1)  /* FOR THE POSIBLE DUMMY FEATURE */

/* NEW TYPES DEFINITION */

/* node1_t IS USED FOR SINGLE-LINKED LISTS */
typedef struct node1_t {
  int i;
  double val;
  struct node1_t *Next;
} node1_t;

/* node1_t IS USED FOR DOUBLE-LINKED LISTS */
typedef struct node2_t {
  int i, j;
  double val;
  struct node2_t *NextC;               /* NEXT COLUMN */
  struct node2_t *NextR;               /* NEXT ROW */
} node2_t;



/* GLOBAL VARIABLE DECLARATION */
static int _n1, _n2;                          /* SIGNATURES SIZES */
static float _C[MAX_SIG_SIZE1][MAX_SIG_SIZE1];/* THE COST MATRIX */
static node2_t _X[MAX_SIG_SIZE1*2];            /* THE BASIC VARIABLES VECTOR */
```

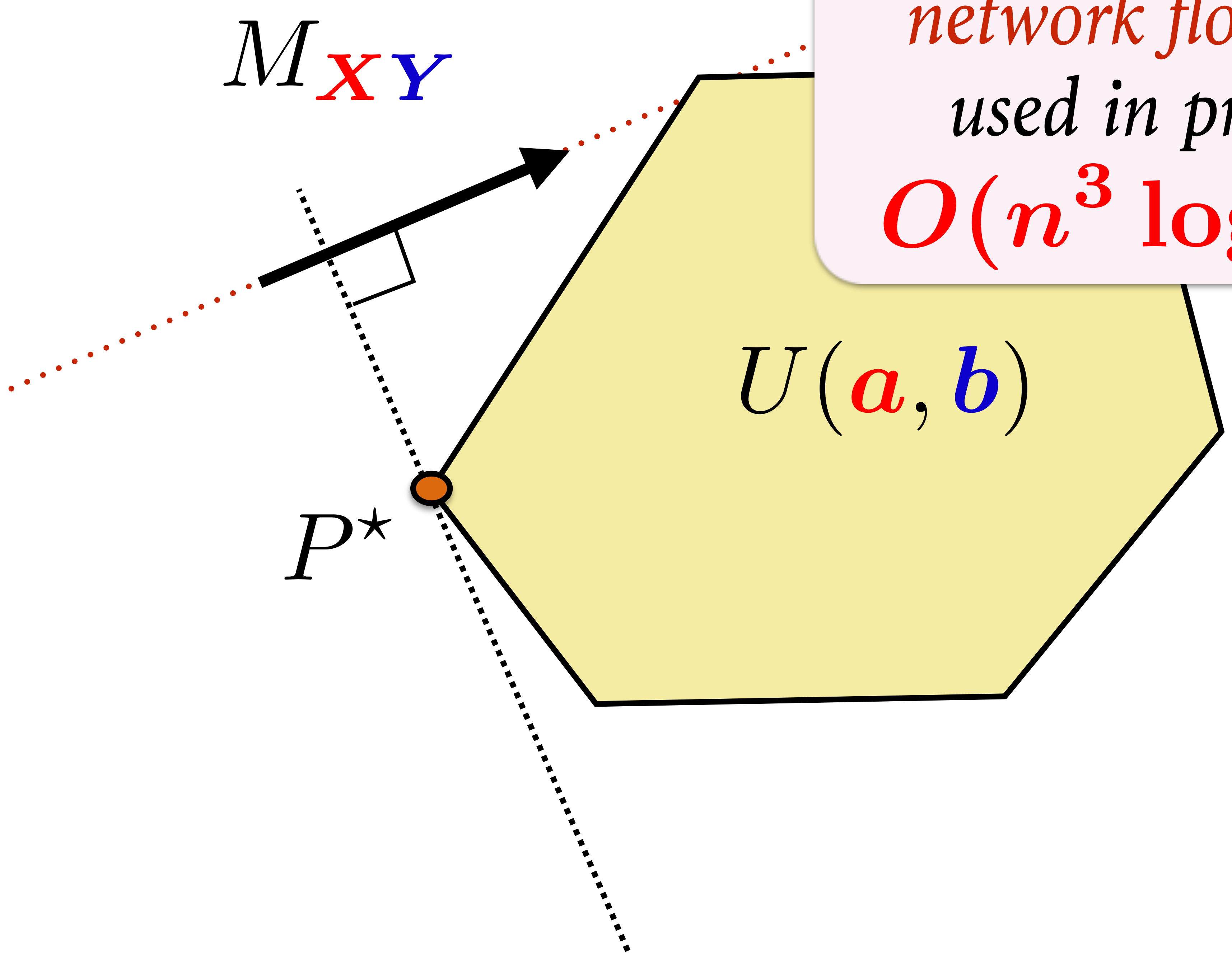
Discrete OT Problem



Discrete OT Problem



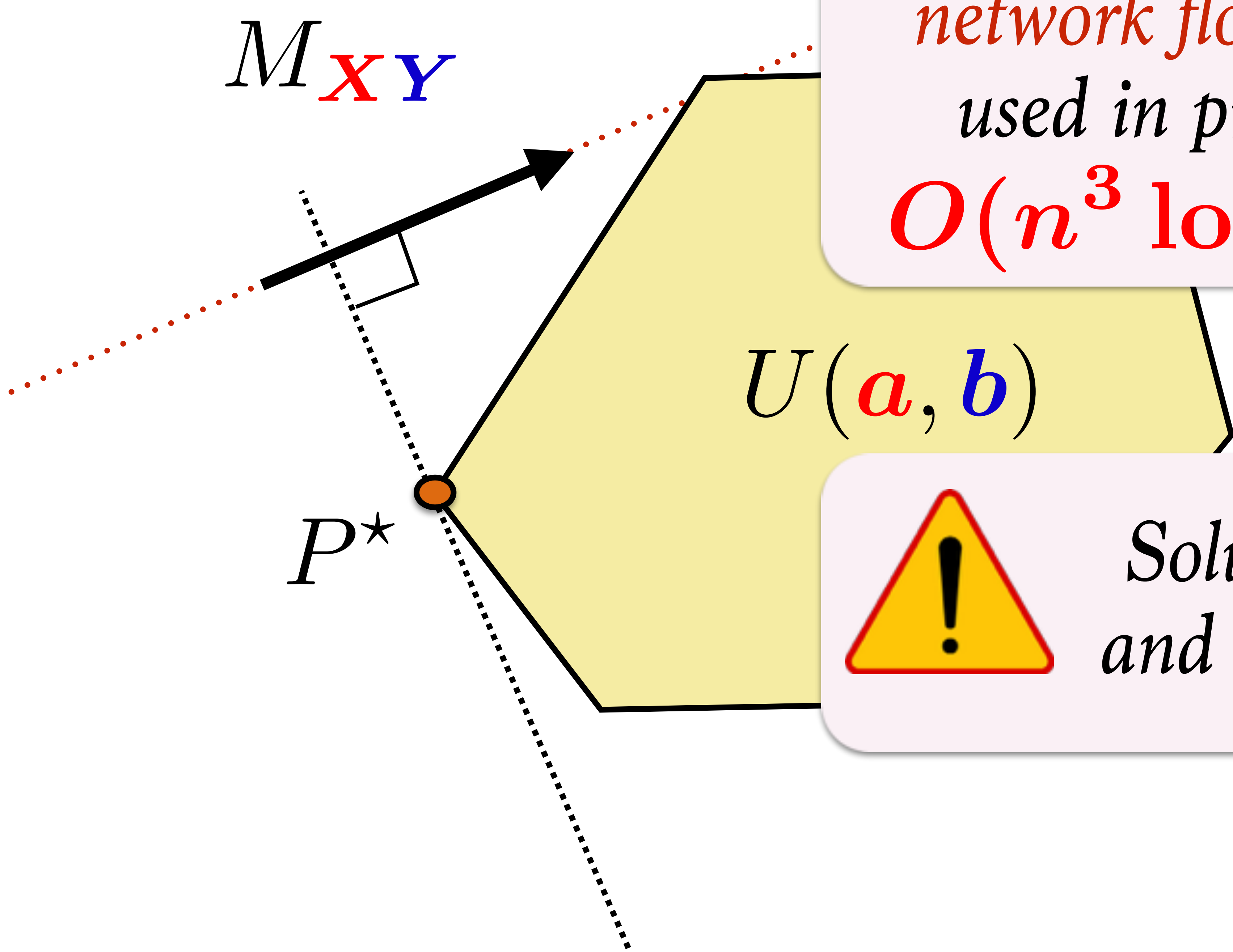
Discrete OT Problem



network flow solver used in practice.

$O(n^3 \log(n))$

Discrete OT Problem

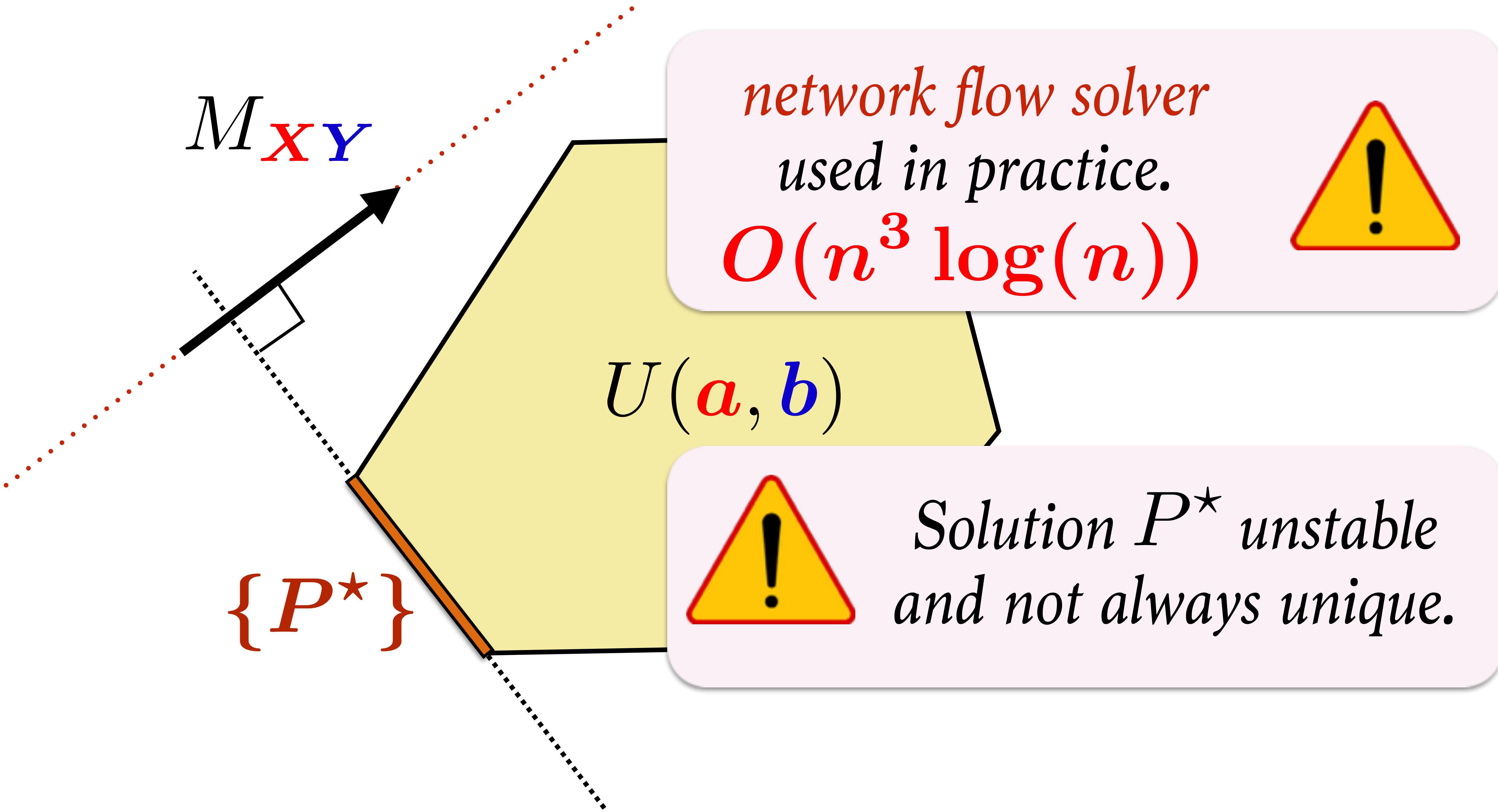


network flow solver *used in practice.*

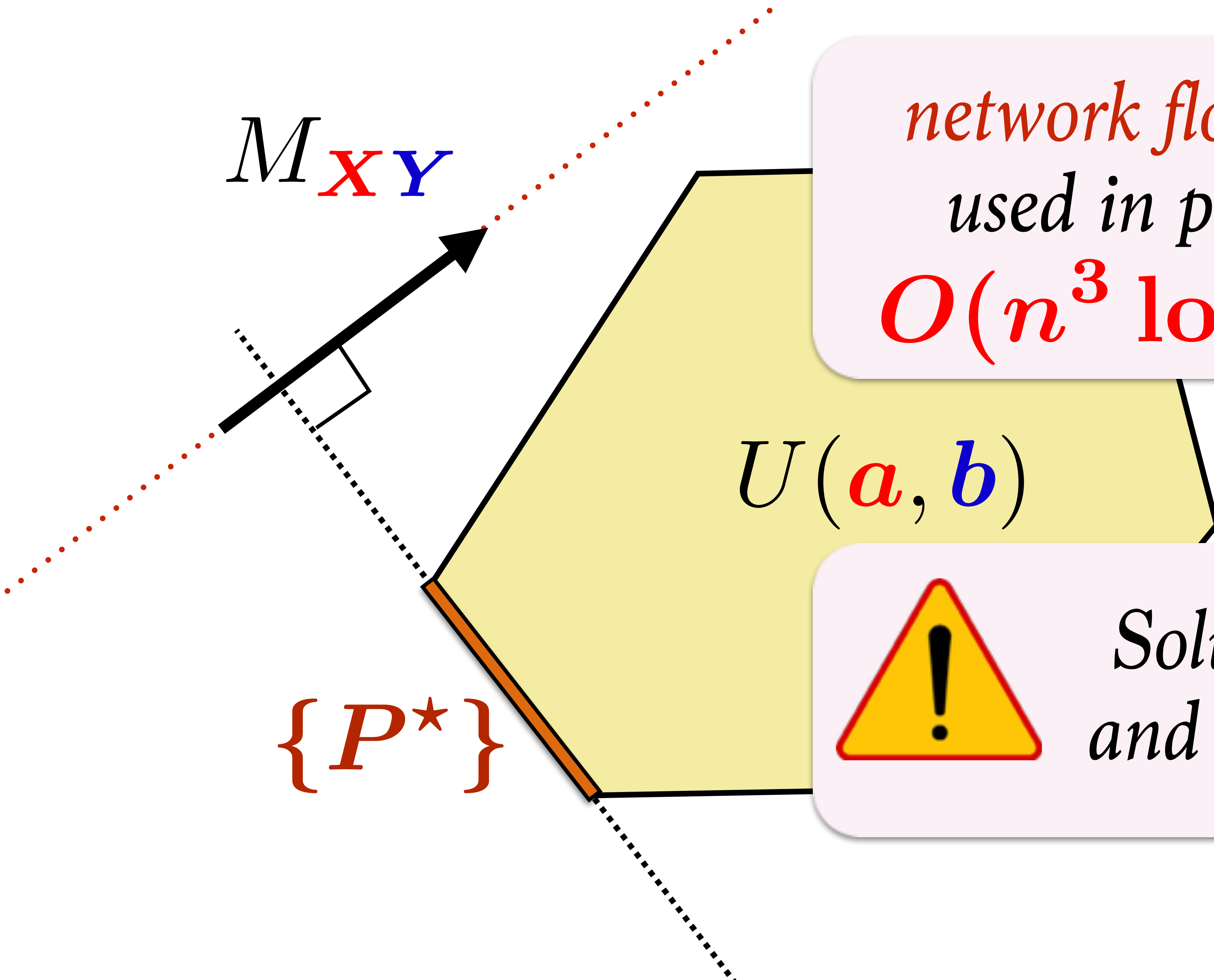
$O(n^3 \log(n))$

Solution $P^\star$ *unstable and not always unique.*

Discrete OT Problem



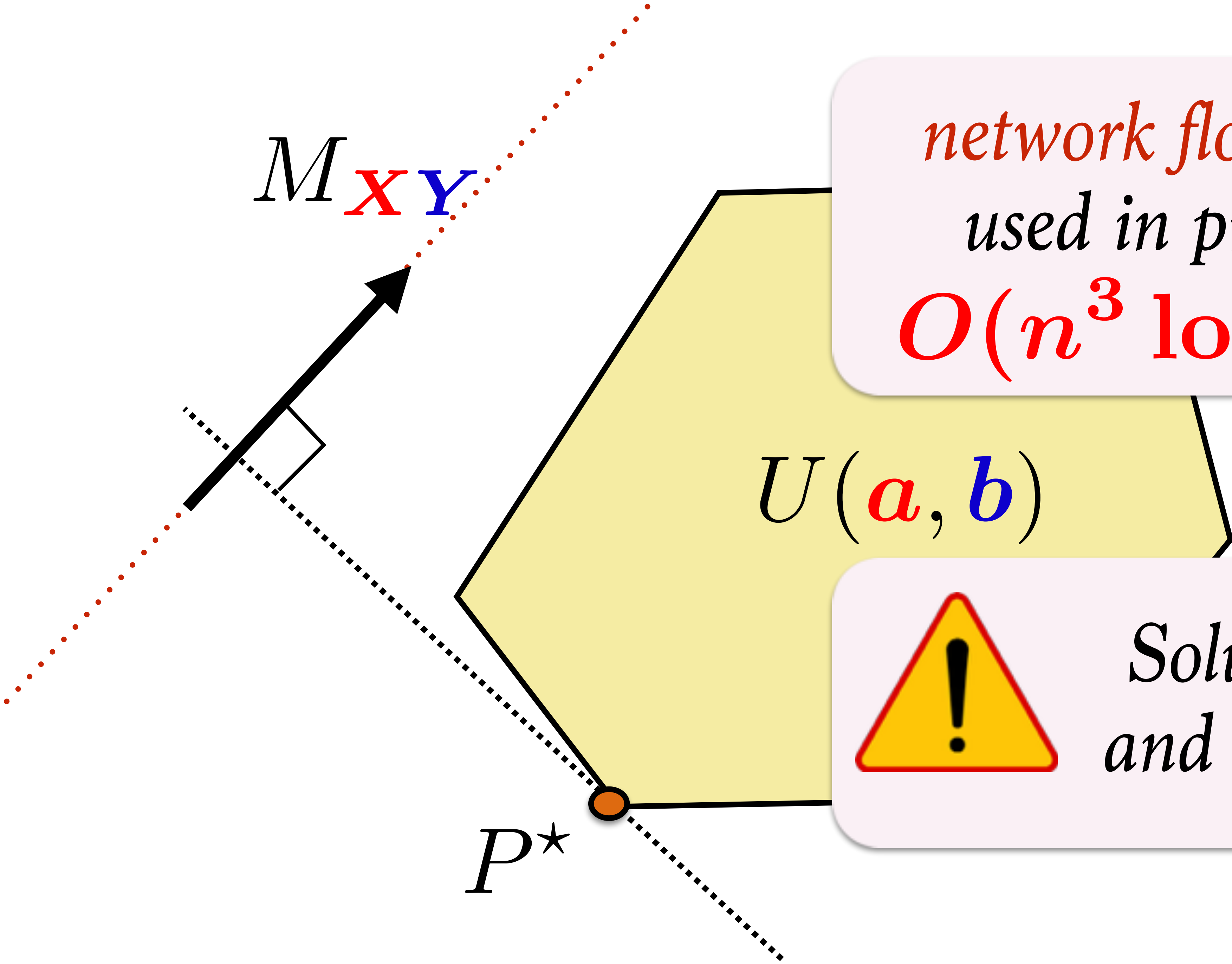
Discrete OT Problem



network flow solver
used in practice.
$O(n^3 \log(n))$

Solution $P^\star$ unstable
and not always unique.

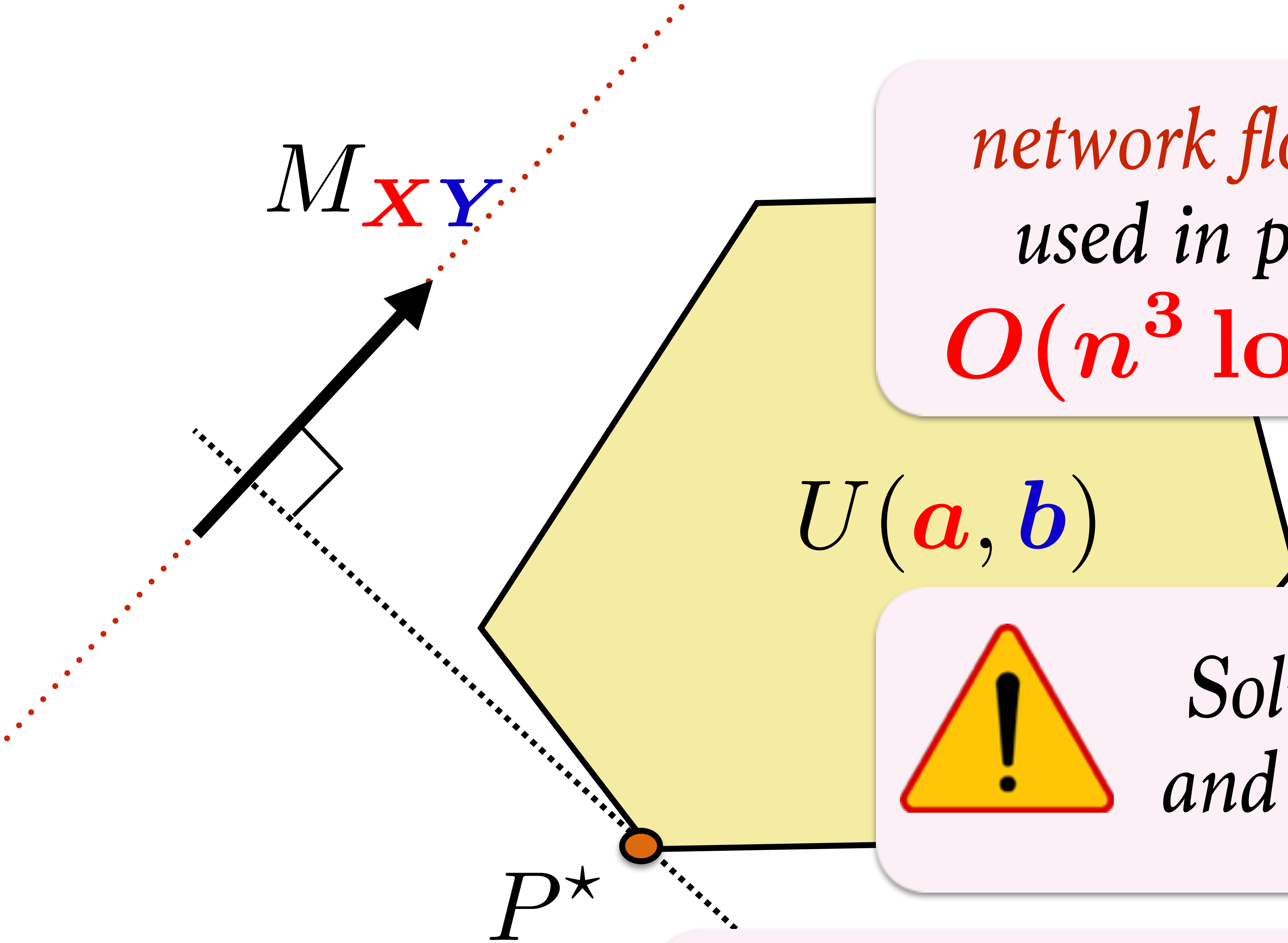
Discrete OT Problem



network flow solver *used in practice.*
$\boldsymbol{O(n^3 \log(n))}$

Solution $P^\star$ *unstable and not always unique.*

Discrete OT Problem



network flow solver
used in practice.
$O(n^3 \log(n))$

Solution $P^\star$ unstable and not always unique.

$W_p^p(\boldsymbol{\mu}, \boldsymbol{\nu})$ not differentiable.

Entropic Regularization [**Wilson'62**]

Def. Regularized Wasserstein, $\gamma \geq 0$

$$W_\gamma(\boldsymbol{\mu}, \boldsymbol{\nu}) \stackrel{\text{def}}{=} \min_{\boldsymbol{P} \in U(\boldsymbol{a}, \boldsymbol{b})} \langle \boldsymbol{P}, M_{\mathbf{XY}} \rangle - \gamma E(\boldsymbol{P})$$

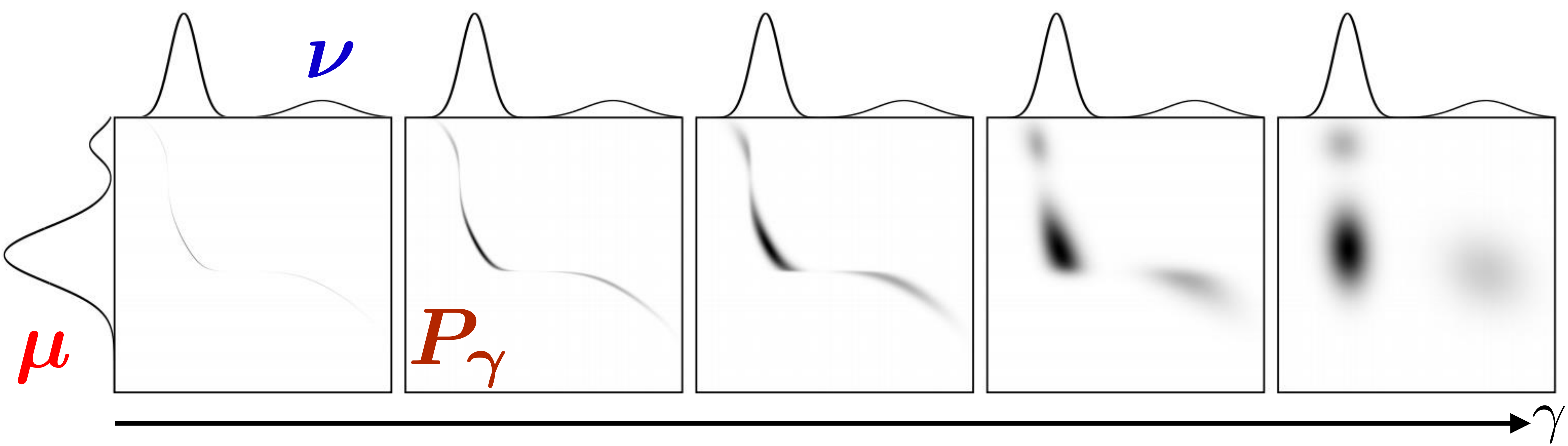
$$E(P) \stackrel{\text{def}}{=} - \sum_{i,j=1}^{nm} P_{ij} (\log P_{ij})$$

Note: Unique optimal solution because of strong concavity of Entropy

Entropic Regularization [Wilson'62]

Def. Regularized Wasserstein, $\gamma \geq 0$

$$W_\gamma(\boldsymbol{\mu}, \boldsymbol{\nu}) \stackrel{\text{def}}{=} \min_{\boldsymbol{P} \in U(\boldsymbol{a}, \boldsymbol{b})} \langle \boldsymbol{P}, M_{\boldsymbol{XY}} \rangle - \gamma E(\boldsymbol{P})$$



Note: Unique optimal solution because of strong concavity of Entropy

Fast & Scalable Algorithm

Prop. If $P_\gamma \stackrel{\text{def}}{=} \underset{\boldsymbol{P}\in U(\boldsymbol{a},\boldsymbol{b})}{\operatorname{argmin}} \langle \boldsymbol{P}, M_{\mathbf{XY}} \rangle - \gamma E(\boldsymbol{P})$

then $\exists! \boldsymbol{u} \in \mathbb{R}_+^n, \boldsymbol{v} \in \mathbb{R}_+^m$, such that

$$P_\gamma = \mathbf{diag}(\boldsymbol{u}) K \mathbf{diag}(\boldsymbol{v}), \quad K \stackrel{\text{def}}{=} e^{-M_{\mathbf{XY}}/\gamma}$$

Fast & Scalable Algorithm

Prop. If $P_\gamma \stackrel{\text{def}}{=} \operatorname*{argmin}_{\boldsymbol{P} \in U(\boldsymbol{a},\boldsymbol{b})} \langle \boldsymbol{P}, M_{\mathbf{XY}} \rangle - \gamma E(\boldsymbol{P})$

then $\exists! \boldsymbol{u} \in \mathbb{R}_+^n, \boldsymbol{v} \in \mathbb{R}_+^m$, such that

$$P_\gamma = \mathbf{diag}(\boldsymbol{u}) K \mathbf{diag}(\boldsymbol{v}), \quad K \stackrel{\text{def}}{=} e^{-M_{\mathbf{XY}}/\gamma}$$

$$L(P, \alpha, \beta) = \sum_{ij} P_{ij} M_{ij} + \gamma P_{ij} \log P_{ij} + \alpha^T (P\mathbf{1} - \boldsymbol{a}) + \beta^T (P^T \mathbf{1} - \boldsymbol{b})$$

$$\partial L / \partial P_{ij} = M_{ij} + \gamma (\log P_{ij} + 1) + \alpha_i + \beta_j$$

$$(\boldsymbol{\partial L / \partial P_{ij}} = 0) \Rightarrow P_{ij} = e^{\frac{\alpha_i}{\gamma} + \frac{1}{2}} e^{-\frac{M_{ij}}{\gamma}} e^{\frac{\beta_j}{\gamma} + \frac{1}{2}} = \boldsymbol{u_i} K_{ij} \boldsymbol{v_j}$$

Fast & Scalable Algorithm

Prop. If $P_\gamma \stackrel{\text{def}}{=} \underset{\boldsymbol{P} \in U(\boldsymbol{a},\boldsymbol{b})}{\operatorname{argmin}} \langle \boldsymbol{P}, M_{\mathbf{XY}} \rangle - \gamma E(\boldsymbol{P})$

then $\exists! \boldsymbol{u} \in \mathbb{R}^n_+, \boldsymbol{v} \in \mathbb{R}^m_+$, such that

$$P_\gamma = \mathbf{diag}(\boldsymbol{u}) K \mathbf{diag}(\boldsymbol{v}), \quad K \stackrel{\text{def}}{=} e^{-M_{\mathbf{XY}}/\gamma}$$

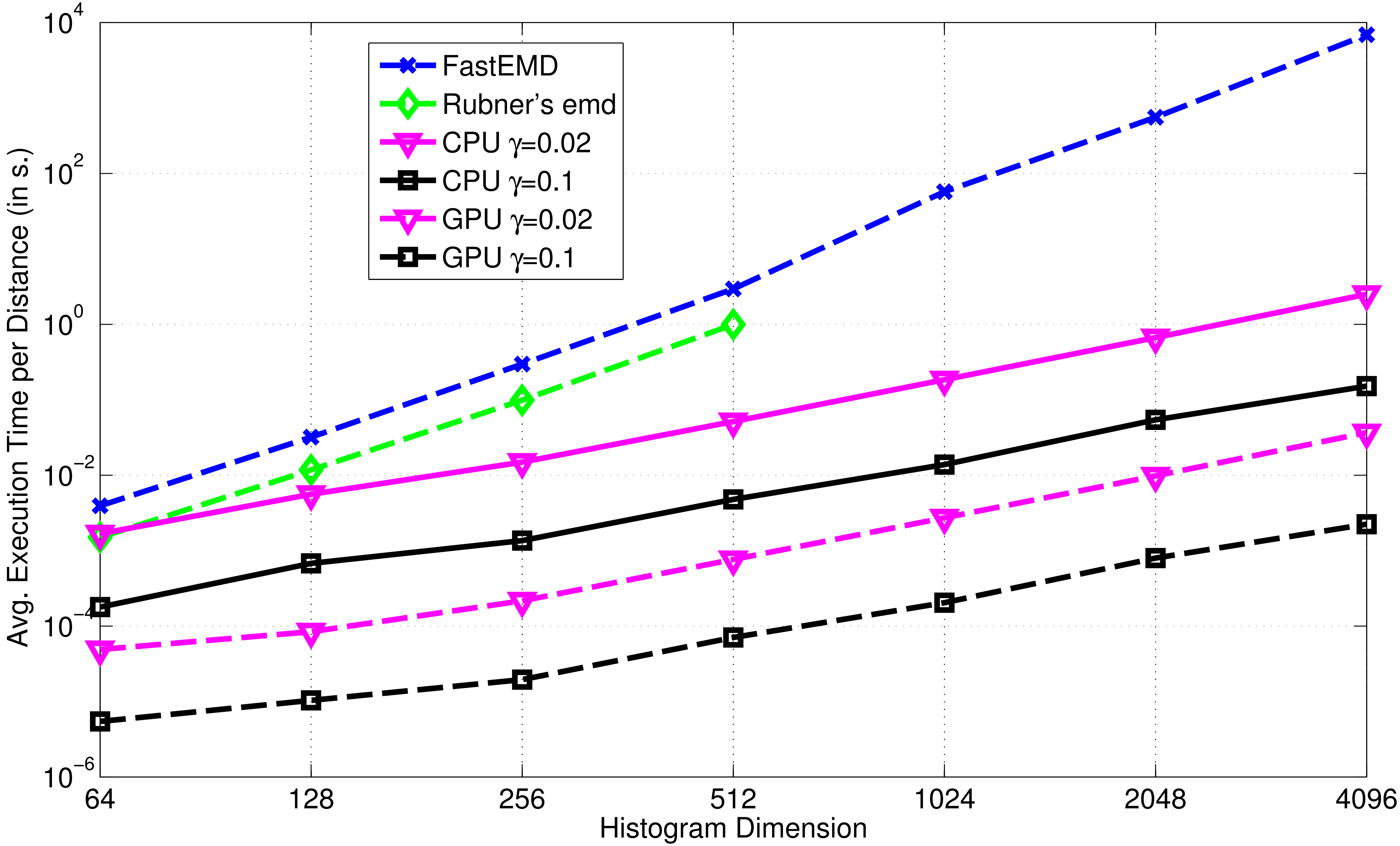
- **[Sinkhorn'64]** fixed-point iterations for $(\boldsymbol{u}, \boldsymbol{v})$

$$\boldsymbol{u} \leftarrow \boldsymbol{a}/K\boldsymbol{v}, \quad \boldsymbol{v} \leftarrow \boldsymbol{b}/K^T\boldsymbol{u}$$

- $O(nm)$ complexity, GPGPU parallel **[C'13]** .
- $O(n^{d+1})$ if $\Omega = \{1, \ldots, n\}^d$ and D^p separable.

[S..C..'15]

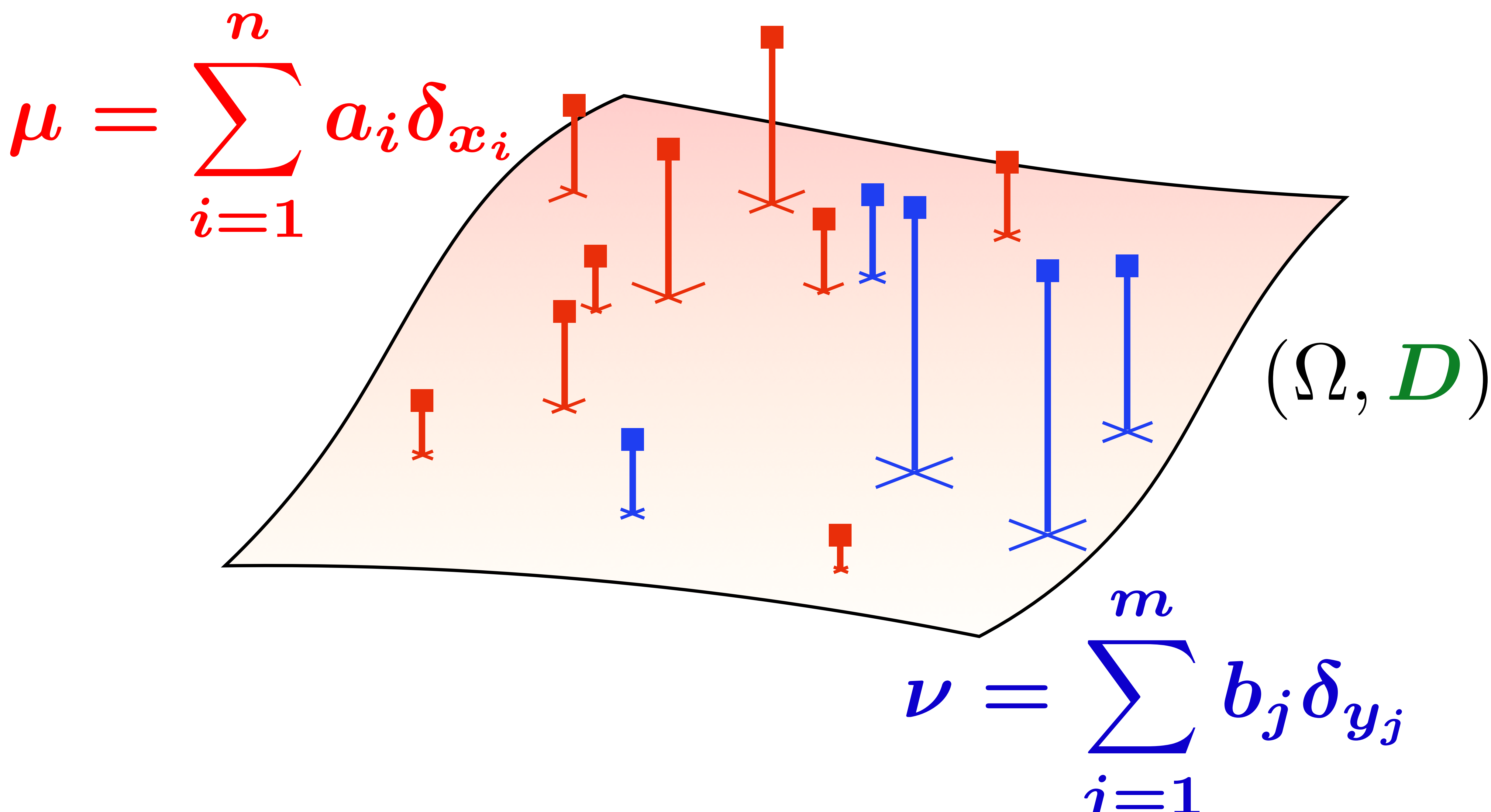
Very Fast EMD Approx. Solver



Note. (Ω, D) is a random graph with shortest path metric, histograms sampled uniformly on simplex, Sinkhorn tolerance 10^{-2}.

Regularization ⇝ *Differentiability*

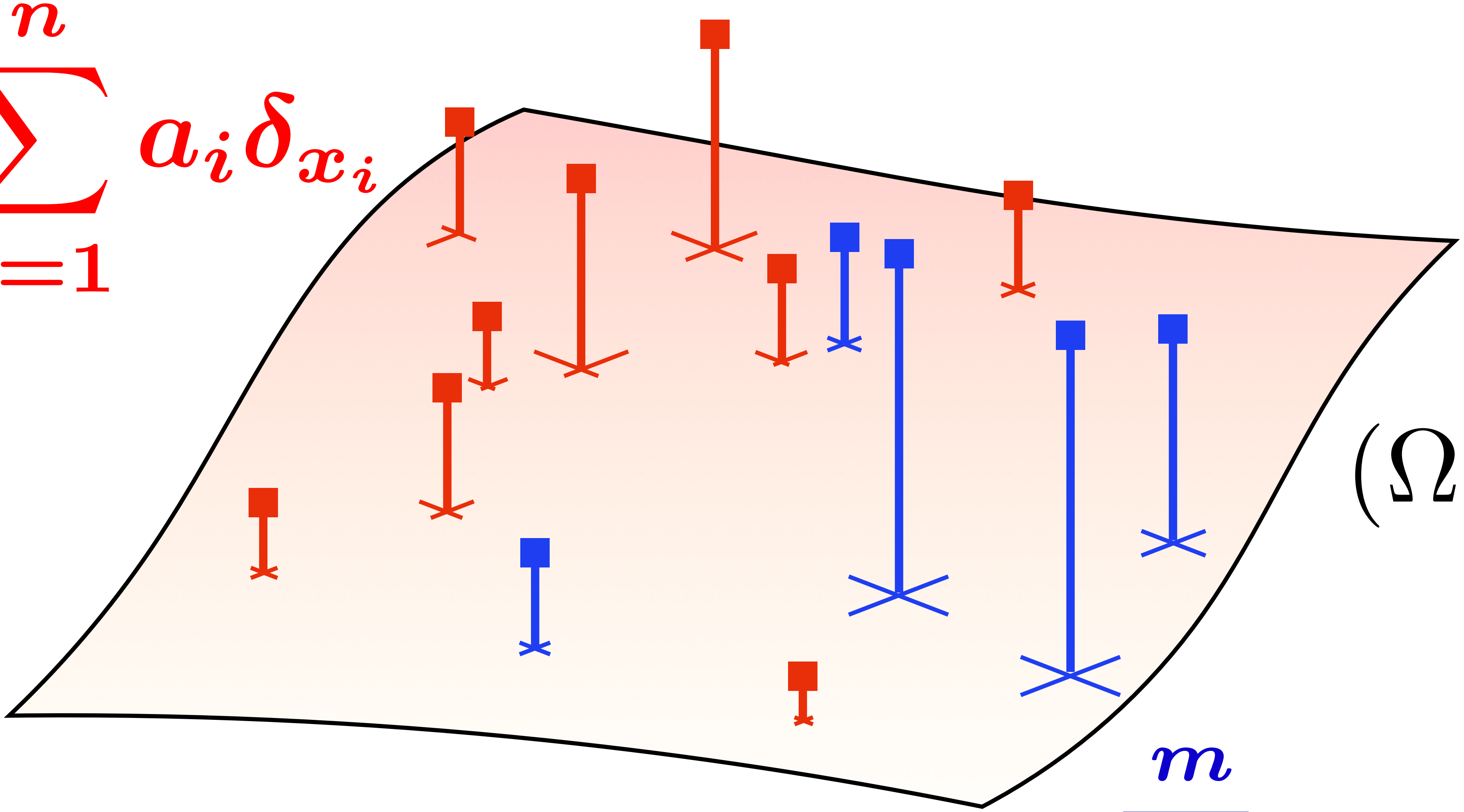
$$W_\gamma((\boldsymbol{a},\boldsymbol{X}),(\boldsymbol{b},\boldsymbol{Y})) = \min_{\boldsymbol{P}\in U(\boldsymbol{a},\boldsymbol{b})} \langle \boldsymbol{P}, M_{\mathbf{XY}} \rangle - \gamma E(\boldsymbol{P})$$



Regularization ⇝ *Differentiability*

$$W_\gamma((\boldsymbol{a} + \boldsymbol{\Delta a}, \boldsymbol{X}), (\boldsymbol{b}, \boldsymbol{Y})) = W_\gamma((\boldsymbol{a}, \boldsymbol{X}), (\boldsymbol{b}, \boldsymbol{Y})) + ??$$

$$\boldsymbol{\mu} = \sum_{i=1}^{n} \boldsymbol{a_i \delta_{x_i}}$$



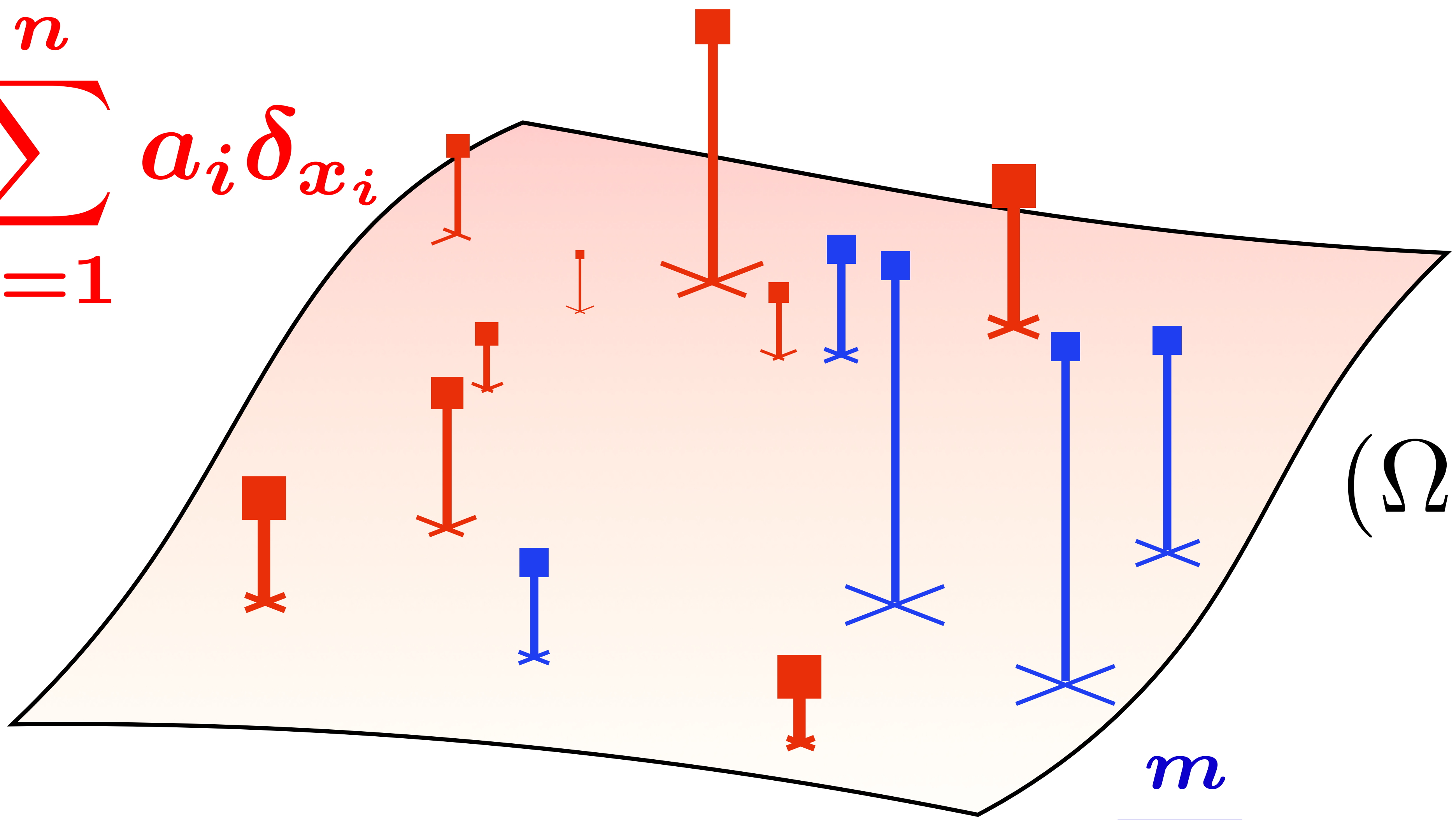
$$(\Omega, \boldsymbol{D})$$

$$\boldsymbol{\nu} = \sum_{j=1}^{m} \boldsymbol{b_j \delta_{y_j}}$$

Regularization ⤳ *Differentiability*

$$W_\gamma((\boldsymbol{a} + \boldsymbol{\Delta a}, \boldsymbol{X}), (\boldsymbol{b}, \boldsymbol{Y})) = W_\gamma((\boldsymbol{a}, \boldsymbol{X}), (\boldsymbol{b}, \boldsymbol{Y})) + ??$$

$$\boldsymbol{\mu} = \sum_{i=1}^{n} \boldsymbol{a_i \delta_{x_i}}$$



$$(\Omega, \boldsymbol{D})$$

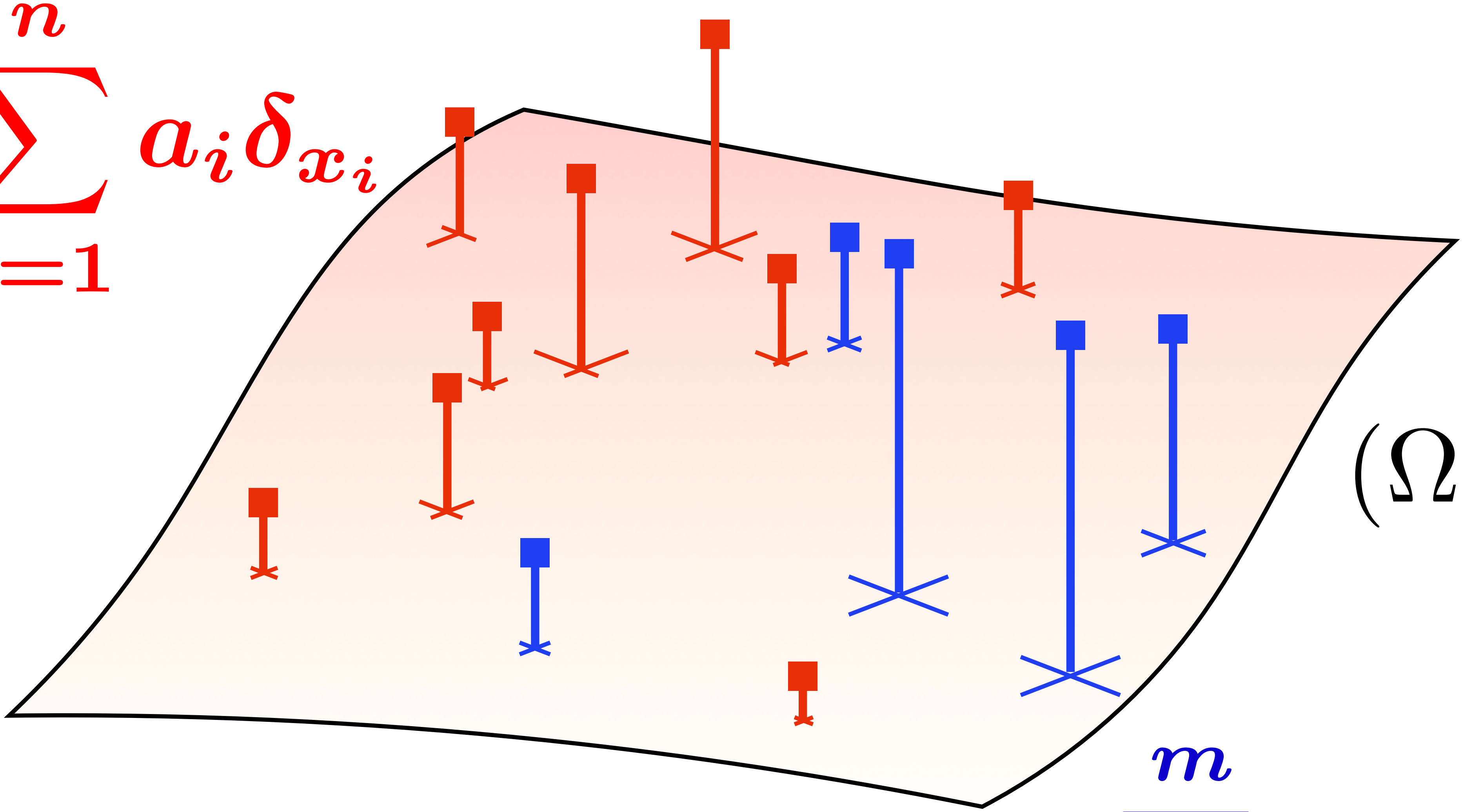
$$\boldsymbol{a} \leftarrow \boldsymbol{a} + \boldsymbol{\Delta a}$$

$$\boldsymbol{\nu} = \sum_{j=1}^{m} \boldsymbol{b_j \delta_{y_j}}$$

Regularization ⇝ *Differentiability*

$$W_\gamma((\boldsymbol{a}, \boldsymbol{X} + \boldsymbol{\Delta X}), (\boldsymbol{b}, \boldsymbol{Y})) = W_\gamma((\boldsymbol{a}, \boldsymbol{X}), (\boldsymbol{b}, \boldsymbol{Y})) + ??$$

$$\boldsymbol{\mu} = \sum_{i=1}^{n} a_i \delta_{x_i}$$



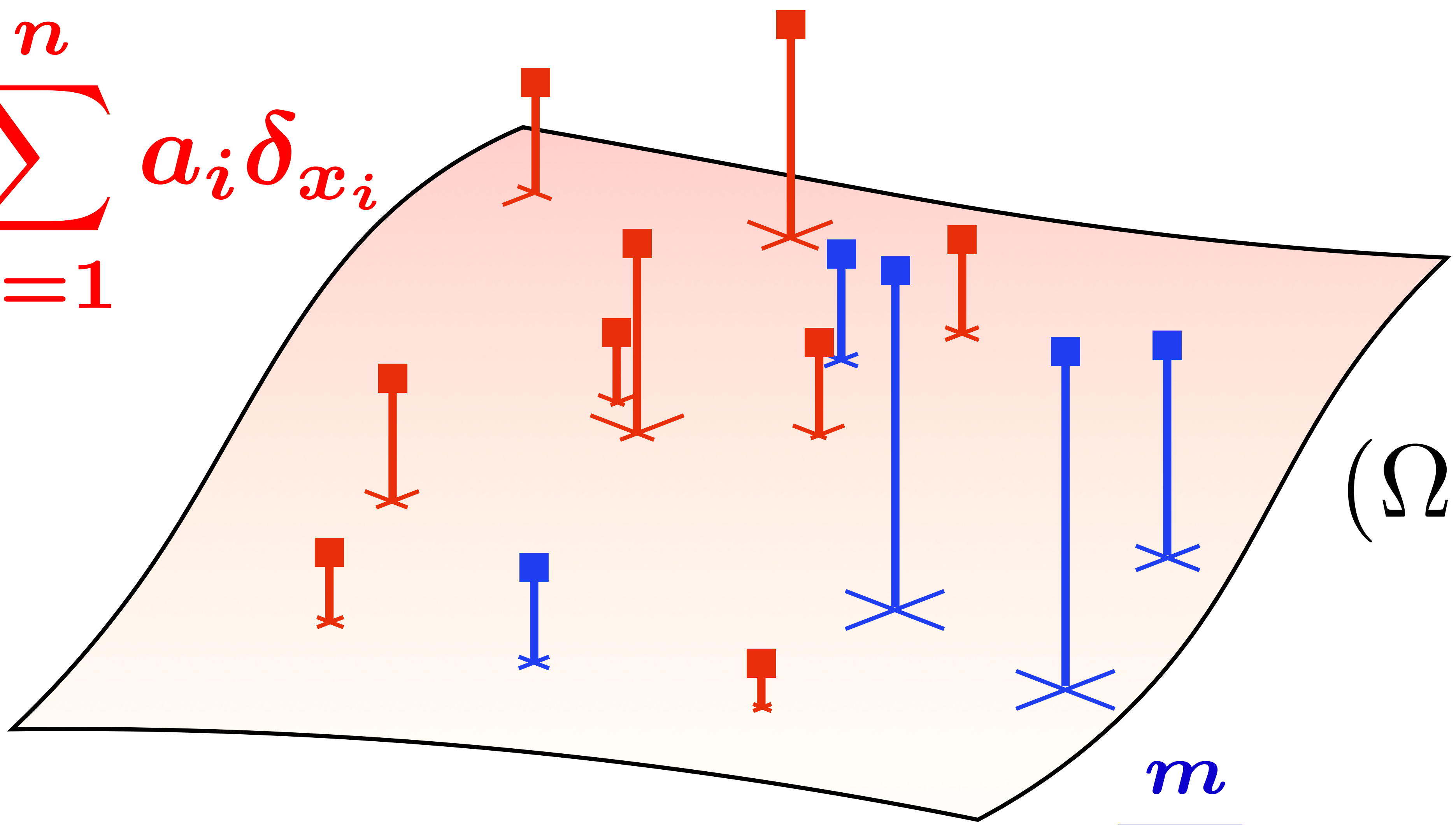
(Ω, D)

$$\boldsymbol{\nu} = \sum_{j=1}^{m} b_j \delta_{y_j}$$

Regularization ⇝ *Differentiability*

$$W_\gamma((\mathbf{a}, \mathbf{X} + \boldsymbol{\Delta}\mathbf{X}), (\mathbf{b}, \mathbf{Y})) = W_\gamma((\mathbf{a}, \mathbf{X}), (\mathbf{b}, \mathbf{Y})) + ??$$

$$\boldsymbol{\mu} = \sum_{i=1}^{n} \boldsymbol{a_i \delta_{x_i}}$$



$$(\Omega, \boldsymbol{D})$$

$$\mathbf{X} \leftarrow \mathbf{X} + \boldsymbol{\Delta}\mathbf{X}$$

$$\boldsymbol{\nu} = \sum_{j=1}^{m} \boldsymbol{b_j \delta_{y_j}}$$

Crucial for "min *data* + *W*" problems

- Quantization, k-means problem **[Lloyd'82]**

$$\min_{\substack{\boldsymbol{\mu}\in\mathcal{P}(\mathbb{R}^d)\\ |\operatorname{supp}\boldsymbol{\mu}|=k}} W_2^2(\boldsymbol{\mu},\boldsymbol{\nu}_{\mathrm{data}})$$

- **[McCann'95]** Interpolant

$$\min_{\boldsymbol{\mu}\in\mathcal{P}(\Omega)} (1-t)W_2^2(\boldsymbol{\mu},\boldsymbol{\nu}_{\mathbf{1}}) + tW_2^2(\boldsymbol{\mu},\boldsymbol{\nu}_{\mathbf{2}})$$

- **[JKO'98]** PDE's as gradient flows in $(\mathcal{P}(\Omega), W)$.

$$\mu_{t+1} = \operatorname*{argmin}_{\boldsymbol{\mu}\in\mathcal{P}(\Omega)} J(\boldsymbol{\mu}) + \lambda_t W_p^p(\boldsymbol{\mu},\mu_t)$$

Crucial for "min *data* + W" problems

Any (ML) problem involving a **KL** or **L2** loss between (parameterized) histograms or probabilility measures can be easily *Wasserstein-ized* if we can differentiate W efficiently.

1. Differentiability of Regularized OT

Def. Dual regularized OT Problem

$$W_\gamma(\boldsymbol{\mu},\boldsymbol{\nu}) = \max_{\alpha,\beta} \alpha^T \boldsymbol{a} + \beta^T \boldsymbol{b} - \frac{1}{\gamma}(e^{\alpha/\gamma})^T \; K e^{\beta/\gamma}$$

Prop. $W_\gamma(\boldsymbol{\mu},\boldsymbol{\nu})$ is **[CD'14]**

1. convex w.r.t. $\boldsymbol{a}$ (Danskin),

$$\nabla_{\boldsymbol{a}} W_\gamma = \alpha^\star = \gamma \log(\boldsymbol{u}).$$

2. decreased, when $p=2, \Omega = \mathbb{R}^d$, using

$$\boldsymbol{X} \leftarrow \boldsymbol{Y} P_\gamma^T \mathbf{D}(\boldsymbol{a}^{-1}).$$

2. Duality for Regularized OT's

[CP'16]

Prop. Writing $H_{\boldsymbol{\nu}} : \boldsymbol{a} \mapsto W_\gamma(\boldsymbol{\mu}, \boldsymbol{\nu})$,

1. $H_{\boldsymbol{\nu}}$ has simple Legendre transform:

$$H^*_{\boldsymbol{\nu}} : \boldsymbol{g} \in \mathbb{R}^n \mapsto \gamma \left(E(\boldsymbol{b}) + \boldsymbol{b}^T \log(K e^{\boldsymbol{g}/\gamma}) \right)$$

2. If $A \in \mathbb{R}^{n \times d}$, f convex on $\mathbb{R}^d$,

$$\min_{\boldsymbol{a} \in \Sigma_n} H_{\boldsymbol{\nu}}(\boldsymbol{a}) + f(A\boldsymbol{a}) = \max_{\boldsymbol{g} \in \mathbb{R}^d} -H^*_{\boldsymbol{\nu}}(A^T \boldsymbol{g}) - f^*(-\boldsymbol{g})$$

3. Stochastic Formulation

$$W_\gamma(\boldsymbol{\mu}, \boldsymbol{\nu}) = \max_{\alpha,\beta} \alpha^T \boldsymbol{a} + \beta^T \boldsymbol{b} - \frac{1}{\gamma}(e^{\alpha/\gamma})^T \; K e^{\beta/\gamma}$$

$$= \max_{\boldsymbol{\alpha}} \boldsymbol{\alpha}^T \boldsymbol{a} - \gamma(\log \; K e^{\boldsymbol{\alpha}/\gamma})^T \boldsymbol{b}$$

$$= \max_{\boldsymbol{\alpha}} \sum_{j=1}^{m} \boldsymbol{b_j} \left(\boldsymbol{\alpha}^T \boldsymbol{a} - \gamma \log \; K_{\cdot \boldsymbol{j}}^T e^{\boldsymbol{\alpha}/\gamma} \right)$$

$$= \max_{\boldsymbol{\alpha}} \sum_{j=1}^{m} \; f_{\boldsymbol{j}}(\boldsymbol{\alpha})$$

- **[GCPB'16]** shows how incremental gradient methods can be used to scale this further.

4. Algorithmic Formulation

Def. For $L \geq 1$, define

$$W_L(\boldsymbol{\mu}, \boldsymbol{\nu}) \stackrel{\text{def}}{=} \langle \boldsymbol{P_L}, M_{\mathbf{XY}} \rangle,$$

where $\boldsymbol{P_L} \stackrel{\text{def}}{=} \mathbf{diag}(\boldsymbol{u_L}) K \mathbf{diag}(\boldsymbol{v_L}),$

$\boldsymbol{v_0} = \mathbf{1}_m; l \geq 0, \boldsymbol{u_l} \stackrel{\text{def}}{=} \boldsymbol{a}/K\boldsymbol{v_l}, \boldsymbol{v_{l+1}} \stackrel{\text{def}}{=} \boldsymbol{b}/K^T\boldsymbol{u_l}.$

Prop. $\frac{\partial W_L}{\partial \mathbf{X}}, \frac{\partial W_L}{\partial \boldsymbol{a}}$ can be computed recursively, in $O(L)$ kernel $K \times$vector products.

Algorithmic Formulation of Reg. OT

Example: Differentiability w.r.t. a

$$\left(\frac{\partial \boldsymbol{v_0}}{\partial a}\right)^T = \mathbf{0}_{m\times n},$$

$$\left(\frac{\partial \boldsymbol{u_l}}{\partial a}\right)^T \boldsymbol{x} = \frac{\boldsymbol{x}}{K\boldsymbol{v_l}} - \left(\frac{\partial \boldsymbol{v_l}}{\partial a}\right)^T K^T \frac{\boldsymbol{x}\circ a}{(K\boldsymbol{v_l})^2},$$

$$\left(\frac{\partial \boldsymbol{v_{l+1}}}{\partial a}\right)^T \boldsymbol{y} = -\left(\frac{\partial \boldsymbol{u_l}}{\partial a}\right)^T K \frac{\boldsymbol{y}\circ b}{(K^T\boldsymbol{u_l})^2}.$$

Algorithmic Formulation of Reg. OT

Example: Differentiability w.r.t. a

$$N = K \circ M_{\mathbf{XY}}$$

$$\nabla_{\boldsymbol{a}} W_L(\boldsymbol{\mu}, \boldsymbol{\nu}) = \left(\frac{\partial \boldsymbol{u_L}}{\partial a}\right)^T N \boldsymbol{v_L} + \left(\frac{\partial \boldsymbol{v_L}}{\partial a}\right)^T N^T \boldsymbol{u_L}$$

```
function [d,grad_a,grad_b,hess_a,hess_b] = sinkhornObjGradHess(a,b,K,M,niter)

u_update = @(v,a) a./(K*v);
v_update = @(u,b) b./(K'*u);

% DuDa = @(eps,dvda,a,v)  (eps./(K*v))- (a./((K*v).^2)).*(K*dvda(eps));
%
% DvDa = @(eps,duda,b,u)  -(b./((K'*u).^2)).*(K'*duda(eps));
%
% DuDb = @(eps,dvdb,a,v)  -(a./((K*v).^2)).*(K*dvdb(eps));
%
% DvDb = @(eps,dudb,b,u)  (eps./(K'*u))-(b./((K'*u).^2)).*(K'*dudb(eps));


DuDat = @(x,dvdat,a,v)  bsxfun(@rdivide,x,K*v)... (x./(K*v))
    -dvdat(K'*( bsxfun(@times,x,(a./((K*v).^2)))));...-dvdat(K'*( (a./((K*v).^2)).*x));


DvDat = @(x,dudat,b,u)  -dudat(K*(bsxfun(@times,x,(b./((K'*u).^2))))); ...(b./((K'*u).^2)).*x))


JDuDat= @(x,Jdvdat,dvdat,a,v) -diag((x'*dvdat(K'))'./((K*v).^2)) ...(K*dvda(x))
    - Jdvdat(x)*K'*diag(a./((K*v).^2))...
    - dvdat(K'* ...
    ( diag(a.*( (-2*(x'*dvdat(K'))')./((K*v).^3)))+...
    diag(x./((K*v).^2))  ));          %1

JDvDat = @(x,Jdudat,dudat,b,u) ...
    -Jdudat(x)*K*diag(b./((K'*u).^2))...
    - dudat(K)* ( ...
    diag(b.*( (-2* (x'*dudat(K))')./((K'*u).^3)))) ;...
```

```
DuDbt = @(x,dvdbt,a,v)  -dvdbt(K'*(bsxfun(@times,x,(a./((K*v).^2))))); ...(a./((K*v).^2)).*x));

DvDbt = @(x,dudbt,b,u)  bsxfun(@rdivide,x,K'*u) ... (x./(K'*u))...
    -dudbt(K*( bsxfun(@times,x,(b./((K'*u).^2)))));...( b./((K'*u).^2)) .*x));




JDvDbt= @(x,Jdudbt,dudbt,b,u) -diag((x'*dudbt(K))'./((K'*u).^2)) ...  (K'*dudb(x))
    - Jdudbt(x)*K*diag(b./((K'*u).^2))...
    - dudbt(K)* ( ...
    diag(b.*( (-2*(x'*dudbt(K))')./((K'*u).^3)))+...
    diag(x./((K'*u).^2))   ) ;

JDuDbt = @(x,Jdvdbt,dvdbt,a,v) ...
    -Jdvdbt(x)*K'*diag(a./((K*v).^2))...
    - dvdbt(K')* ( ...
    diag(a.*( (-2* (x'*dvdbt(K'))')./((K*v).^3)))) ;
```

```
n=size(a,1);
m=size(b,1);

DVDAT= @(eps) zeros(n,size(eps,2));
DVDBT= @(eps) zeros(m,size(eps,2));

JDVDAT= @(eps) zeros(n,m);
JDVDBT= @(eps) zeros(m,m);

v=ones(m,size(b,2));

for j=1:niter,
    u=u_update(v,a);
    DUDAT = @(x) DuDat(x,DVDAT,a,v);
    DUDBT = @(x) DuDbt(x,DVDBT,a,v);

    if nargout>3
        JDUDAT = @(x) JDuDat(x,JDVDAT,DVDAT,a,v);
        JDUDBT = @(x) JDuDbt(x,JDVDBT,DVDBT,a,v);
    end


    v=v_update(u,b);
    DVDAT = @(x) DvDat(x,DUDAT,b,u);
    DVDBT = @(x) DvDbt(x,DUDBT,b,u);

    if nargout>3
        JDVDAT = @(x) JDvDat(x,JDUDAT,DUDAT,b,u);
        JDVDBT = @(x) JDvDbt(x,JDUDBT,DUDBT,b,u);
    end
end
```

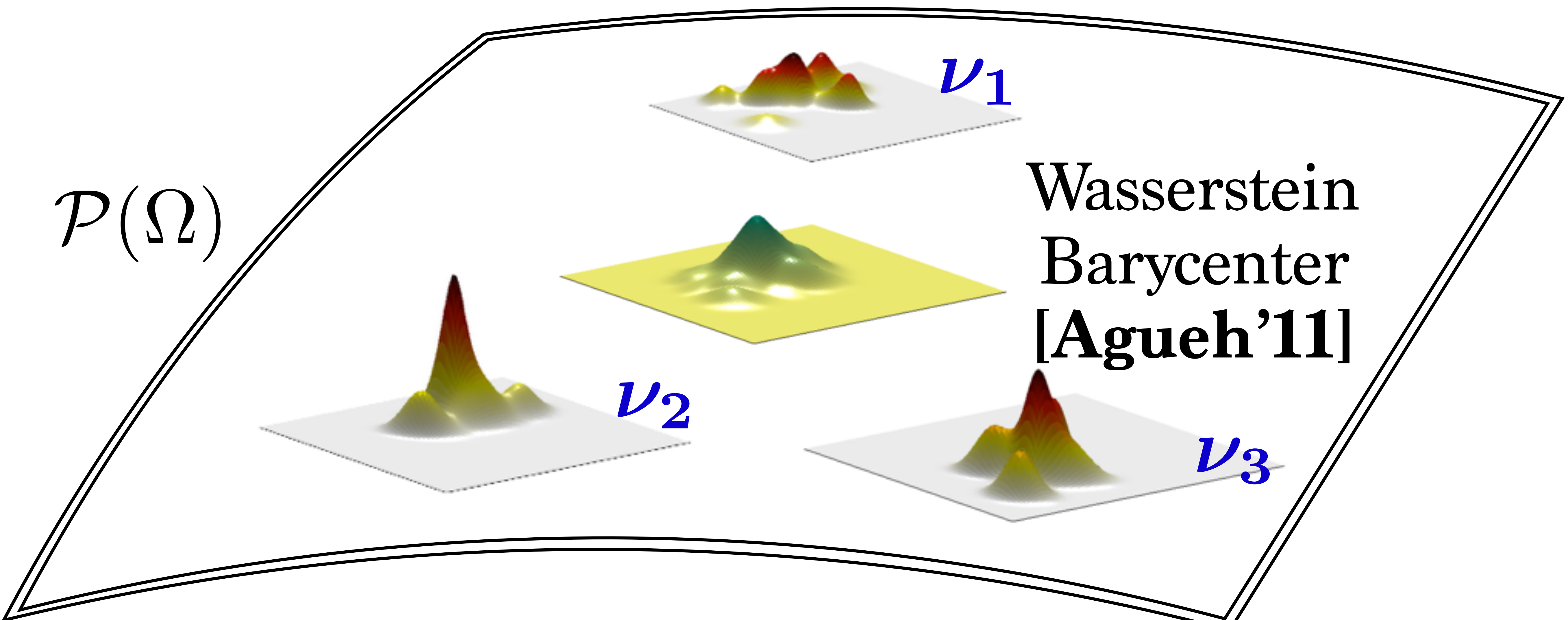
```
U=K.*M;
d=diag(u'*U*v);

grad_a=(DUDAT(U*v)+DVDAT(U'*u));
grad_b=(DUDBT(U*v)+DVDBT(U'*u));


if nargout>3
    hess_a= @(eps) JDUDAT(eps)*(U*v)+DUDAT((eps'*DVDAT(U'))')+...
        JDVDAT(eps)*(U'*u)+DVDAT((eps'*DUDAT(U))');
end
```

Thanks to these tricks...

- **[Agueh'11]** Barycenters **[CD'14][BCCNP'15][GCP'15][S..C..'15]**
- **[Burger'12]** TV gradient flow using duality **[CP'16]**
- Dictionary Learning / Latent Factors **[RCP'16]**
- **[Bigot'15]** *W*-PCA **[SC'15]**
- Density fitting / parameter estimation **[MMC'16]**
- Inverse problems / Wasserstein regression **[BPC'16]**

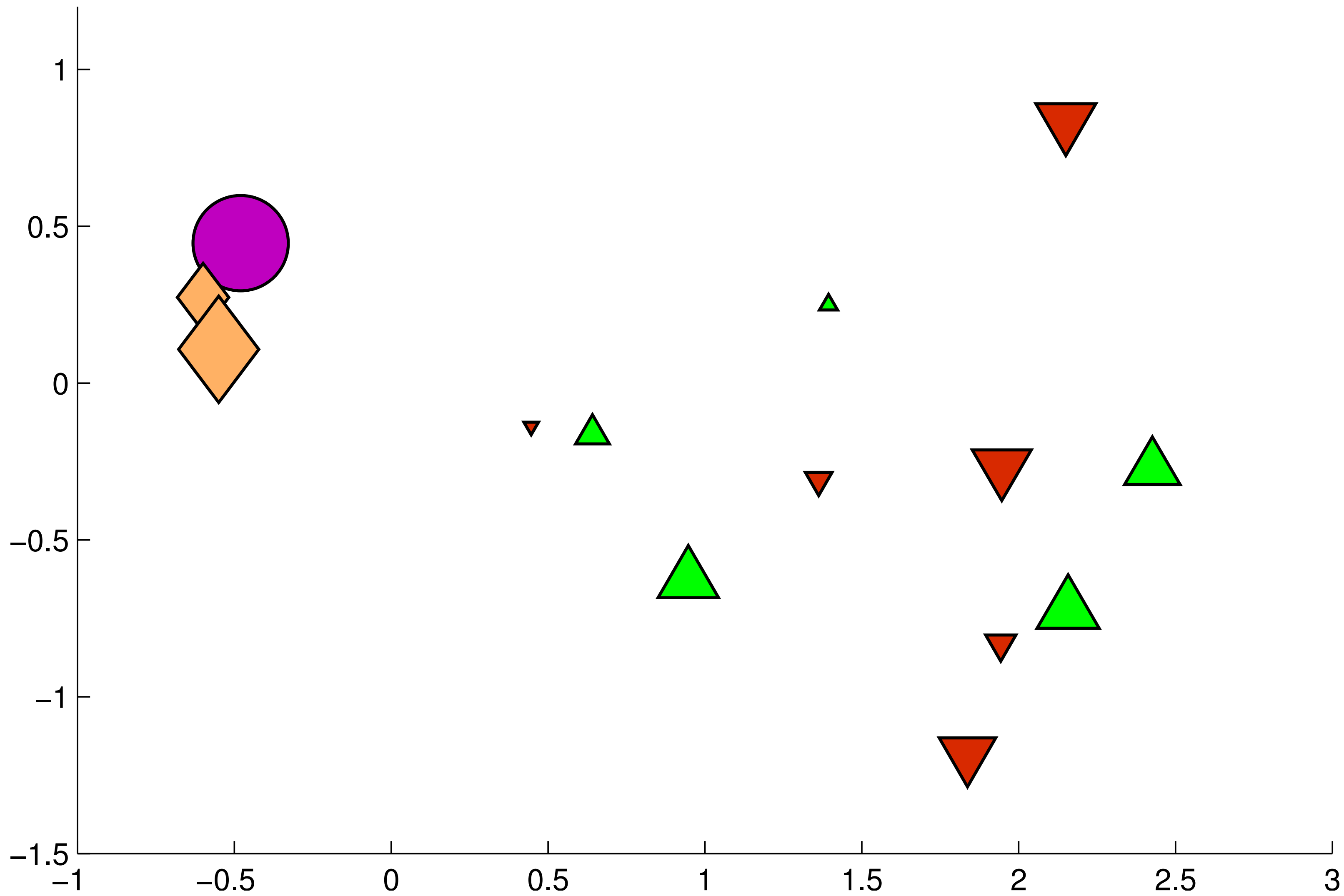
Wasserstein Barycenters

$$\min_{\mu \in \mathcal{P}(\Omega)} \sum_{i=1}^{N} \lambda_i W_p^p(\mu, \nu_i)$$



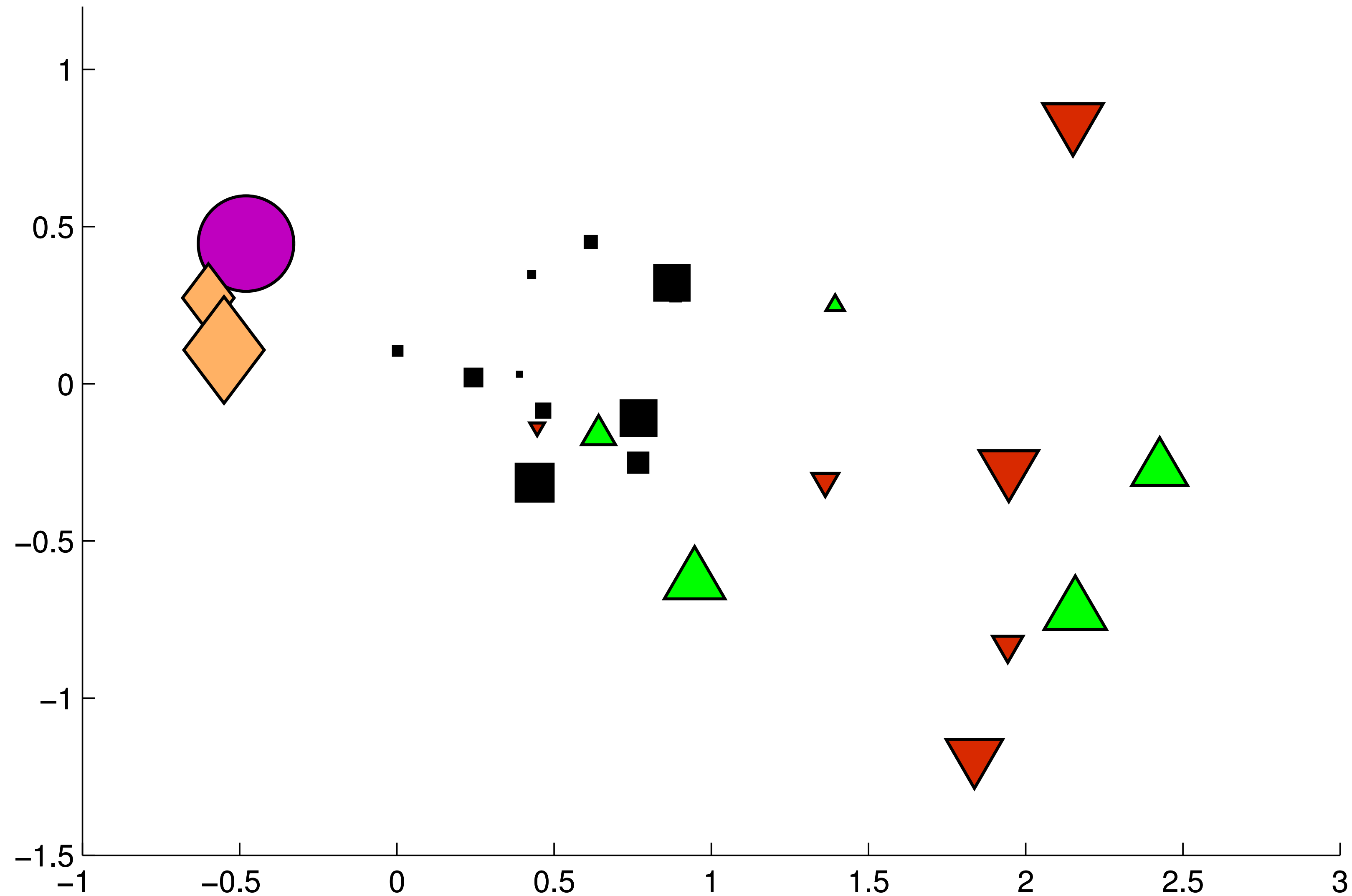
Multimarginal Formulation

- Exact solution (W_2) using MM-OT. **[Agueh'11]**



Multimarginal Formulation

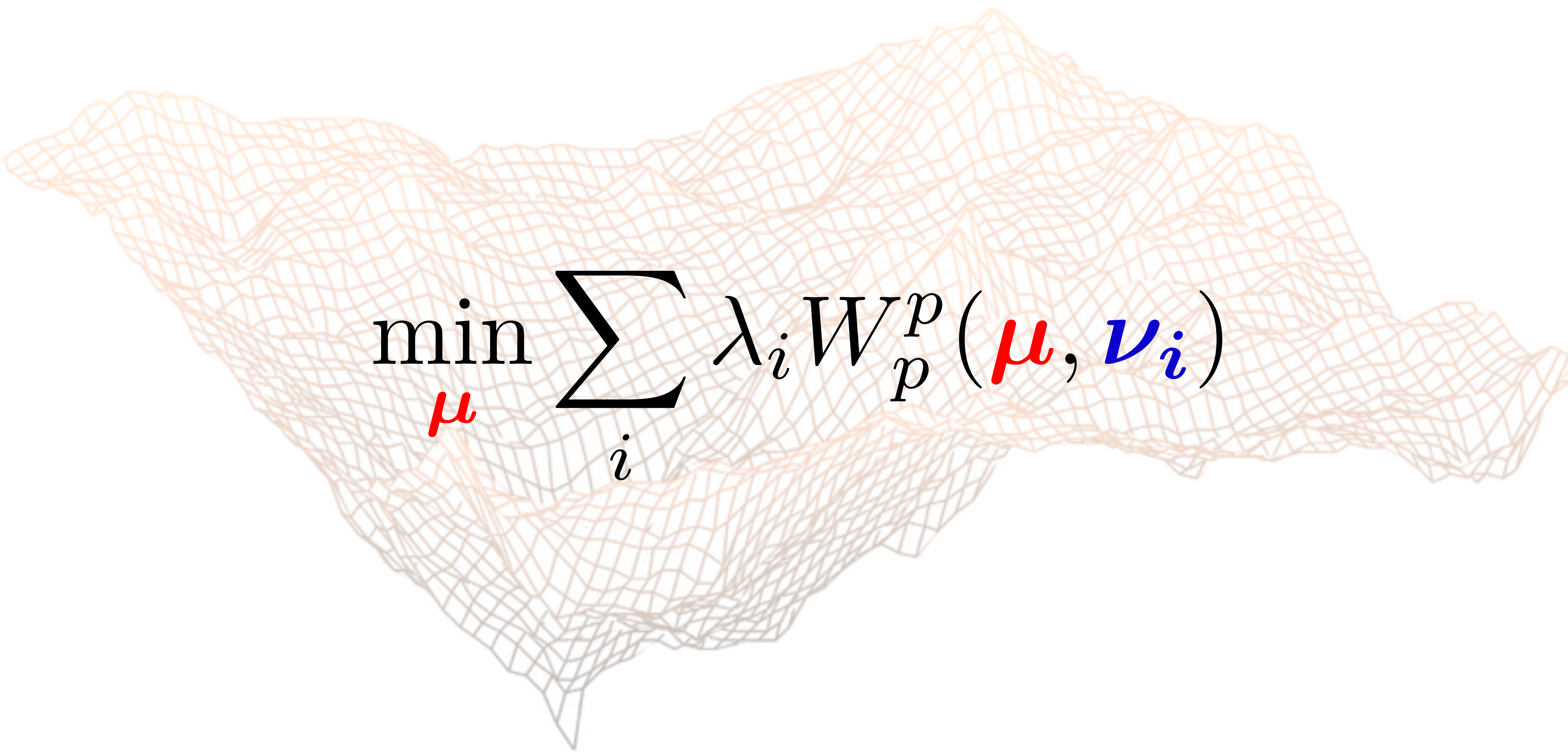
- Exact solution (W_2) using MM-OT. **[Agueh'11]**



$$\text{If } |\operatorname{supp} \boldsymbol{\nu_i}| = \boldsymbol{n_i}, \text{ LP of size } \left(\prod_i \boldsymbol{n_i}, \sum_i \boldsymbol{n_i}\right)$$

Finite Case, LP Formulation

- When Ω is a finite set, metric M, another LP.



$$\min_{\boldsymbol{\mu}} \sum_i \lambda_i W_p^p(\boldsymbol{\mu}, \boldsymbol{\nu_i})$$

Finite Case, LP Formulation

- When Ω is a finite set, metric M, another LP.

$$\min_{\boldsymbol{P_1},\cdots,\boldsymbol{P_N},\boldsymbol{a}} \sum_{i=1}^{N} \lambda_i \langle \boldsymbol{P_i}, M \rangle$$

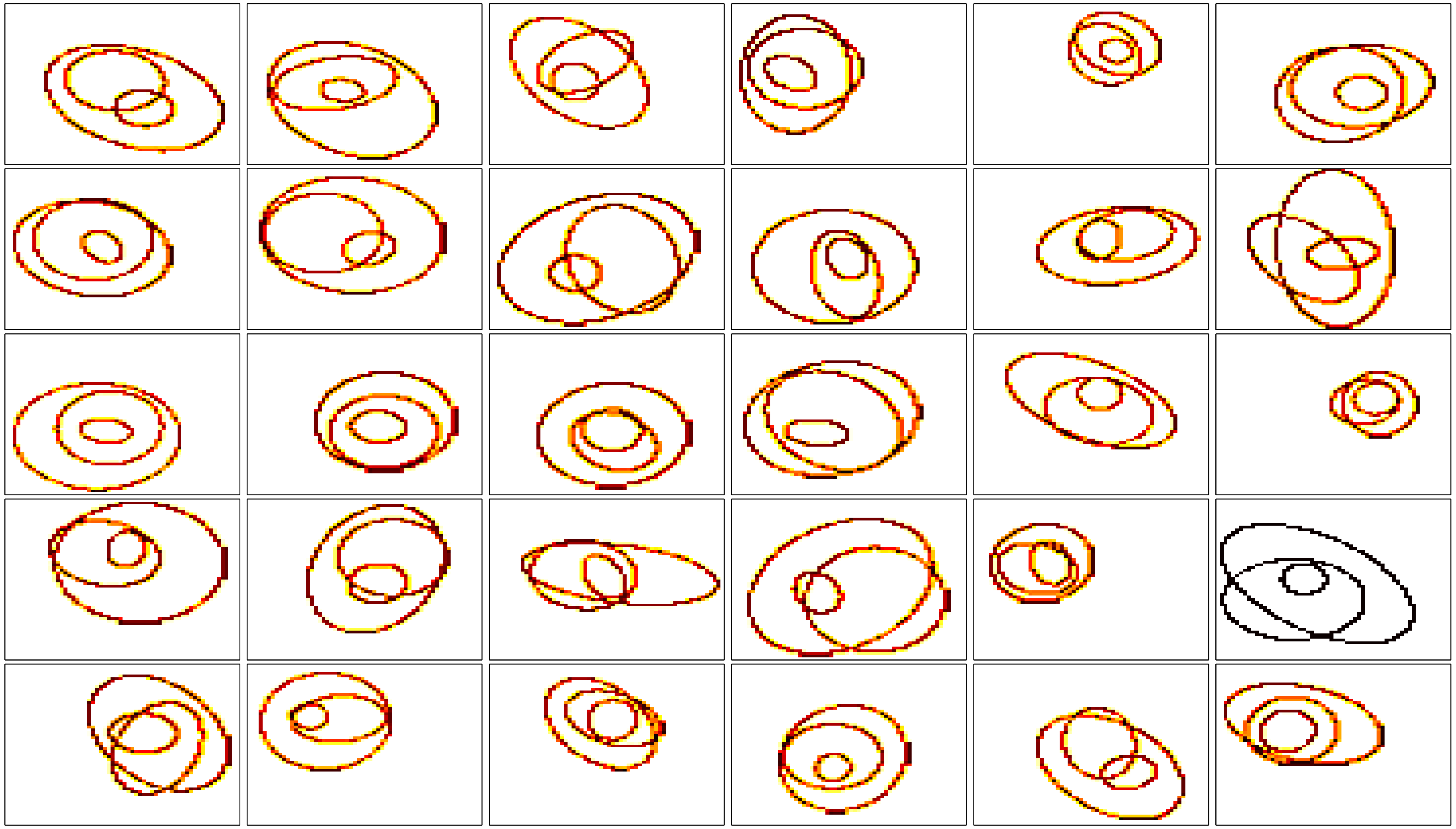
$$\text{s.t.} \quad \boldsymbol{P_i}^T \mathbf{1}_n = \boldsymbol{b_i}, \forall i \leq N,$$

$$\boldsymbol{P_1} \mathbf{1}_n = \cdots = \boldsymbol{P_N} \mathbf{1}_d = \boldsymbol{a}.$$

If $|\Omega| = n$, LP of size $(Nn^2, (2N-1)n)$; unstable

Primal Descent on Regularized W

$$\min_{\mu \in Q \subset \mathcal{P}(\Omega)} \sum_{i=1}^{N} \lambda_i W_{\gamma}(\mu, \nu_i)$$

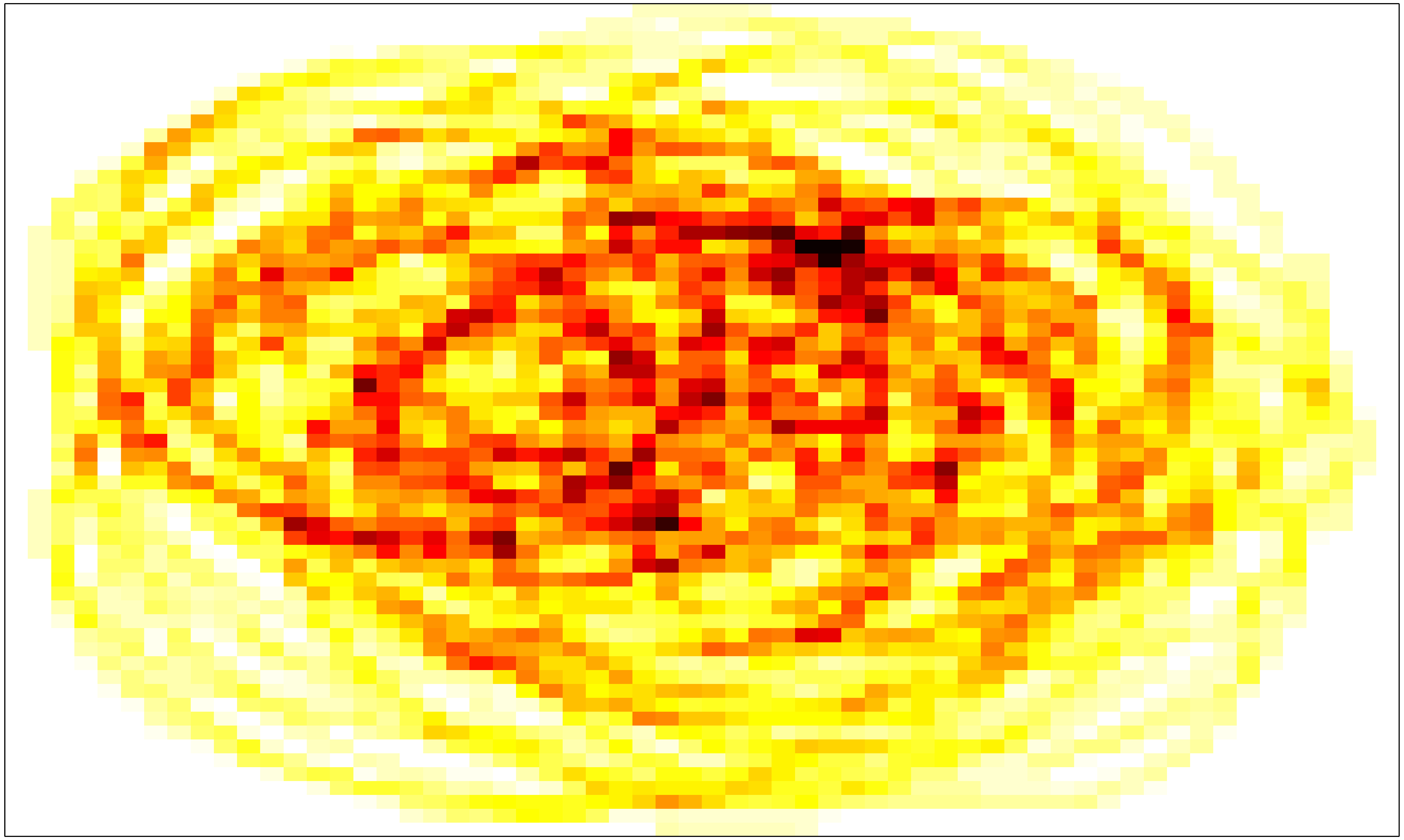


Fast Computation of Wasserstein Barycenters
International Conference on Machine Learning 2014

[CD'14]

Primal Descent on Regularized W

$$\min_{\mu \in Q \subset \mathcal{P}(\Omega)} \sum_{i=1}^{N} \lambda_i W_{\gamma}(\mu, \nu_i)$$

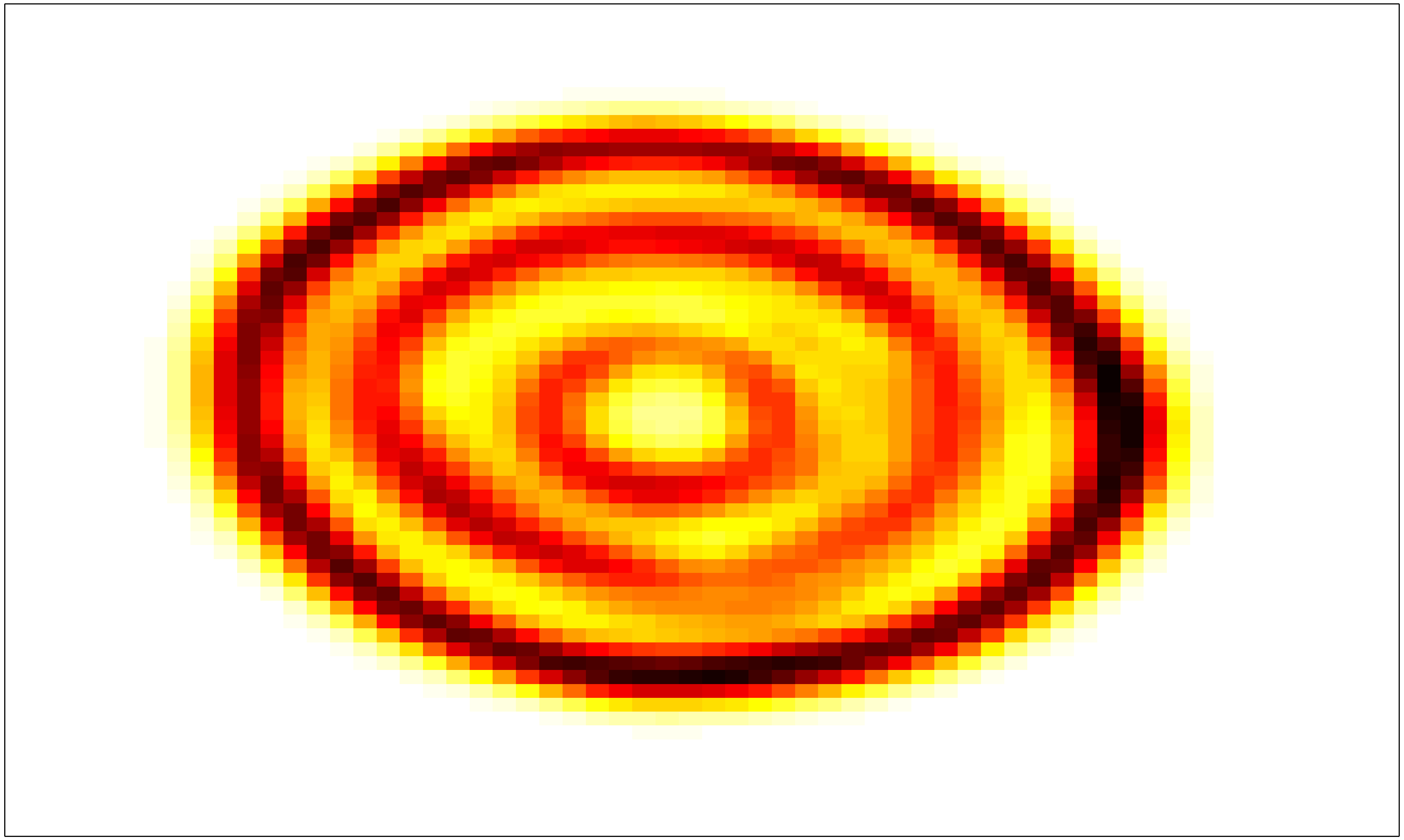


Fast Computation of Wasserstein Barycenters
International Conference on Machine Learning 2014

[CD'14]

Primal Descent on Regularized W

$$\min_{\mu \in Q \subset \mathcal{P}(\Omega)} \sum_{i=1}^{N} \lambda_i W_{\gamma}(\mu, \nu_i)$$



Fast Computation of Wasserstein Barycenters
International Conference on Machine Learning 2014

[CD'14]

Wasserstein Barycenter = KL Projections

$$\langle P, M_{\mathbf{XY}} \rangle - \gamma E(P) = \gamma \mathbf{KL}(P | K)$$

$$\min_{\boldsymbol{a}} \sum_{i=1}^{N} \lambda_i W_\gamma(\boldsymbol{a}, \boldsymbol{b_i}) = \min_{\substack{\mathbf{P}=[P_1,\dots,P_N] \\ \mathbf{P}\in C_1 \cap C_2}} \sum_{i=1}^{N} \lambda_i \mathbf{KL}(\boldsymbol{P_i} | K)$$

$$\boldsymbol{C_1} = \{\mathbf{P} | \exists \boldsymbol{a}, \forall i, P_i \mathbf{1}_m = \boldsymbol{a}\}$$

$$\boldsymbol{C_2} = \{\mathbf{P} | \forall i, P_i^T \mathbf{1}_n = \boldsymbol{b_i}\}$$

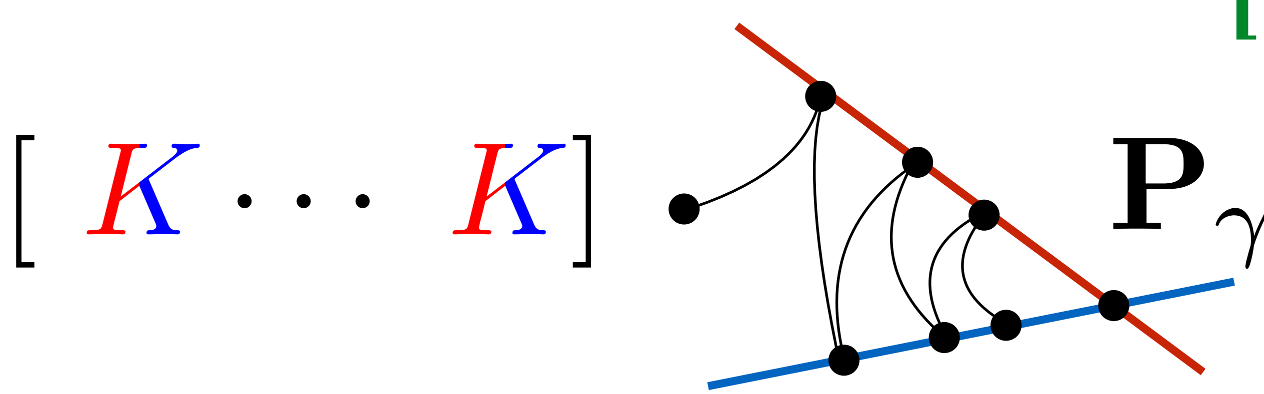
Wasserstein Barycenter = KL Projections

$$\min_{\boldsymbol{a}} \sum_{i=1}^{N} \lambda_i W_\gamma(\boldsymbol{a}, \boldsymbol{b_i}) = \min_{\substack{\mathbf{P}=[P_1,\ldots,P_N] \\ \mathbf{P}\in C_1 \cap C_2}} \sum_{i=1}^{N} \lambda_i \mathbf{KL}(P_i | K)$$

$$C_1 = \{\mathbf{P} | \exists \boldsymbol{a}, \forall i, P_i \mathbf{1}_m = \boldsymbol{a}\}$$

$$C_2 = \{\mathbf{P} | \forall i, P_i^T \mathbf{1}_n = \boldsymbol{b_i}\}$$

[BCCNP'15]



Wasserstein Barycenter = KL Projections

$$\min_{\boldsymbol{a}} \sum_{i=1}^{N} \lambda_i W_\gamma(\boldsymbol{a}, \boldsymbol{b_i}) = \min_{\substack{\mathbf{P}=[P_1,\ldots,P_N] \\ \mathbf{P}\in C_1 \cap C_2}} \sum_{i=1}^{N} \lambda_i \mathbf{KL}(P_i | K)$$

$$C_1 = \{\mathbf{P} | \exists \boldsymbol{a}, \forall i, P_i \mathbf{1}_m = \boldsymbol{a}\}$$

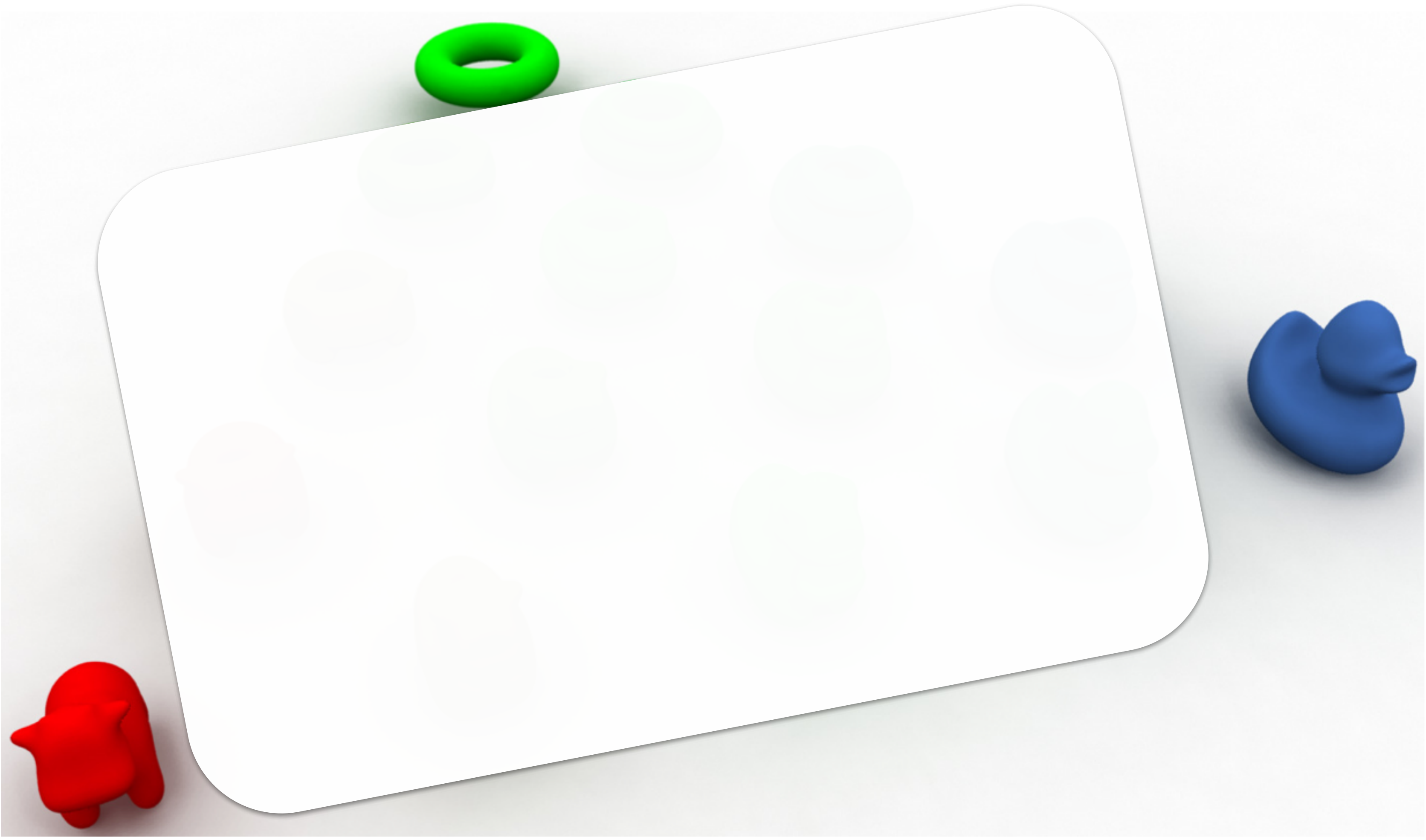
$$C_2 = \left\{\mathbf{P} | \forall i, P_i^T \mathbf{1}_n = \boldsymbol{b_i}\right\}$$

```
u=ones(size(B)); % d x N matrix
while not converged
    v=u.*(K'*(B./(K*u))); % 2(Nd^2) cost
    u=bsxfun(@times,u,exp(log(v)*weights))./v;
end
a=mean(v,2);
```

[BCCNP'15]

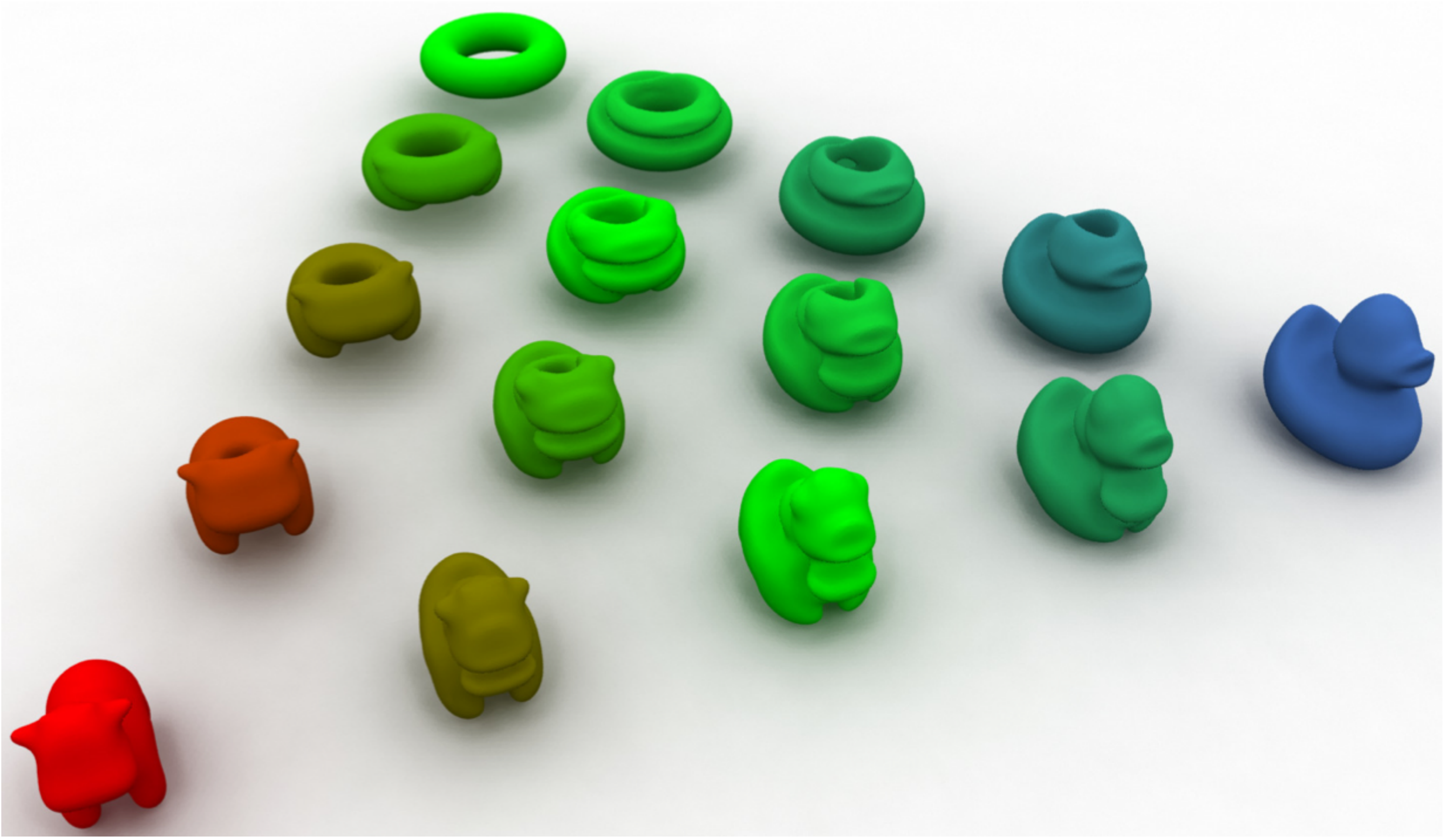
Iterative Bregman Projections for Regularized Transportation Problems
SIAM J. on Sci. Comp. 2015

Application: Graphics



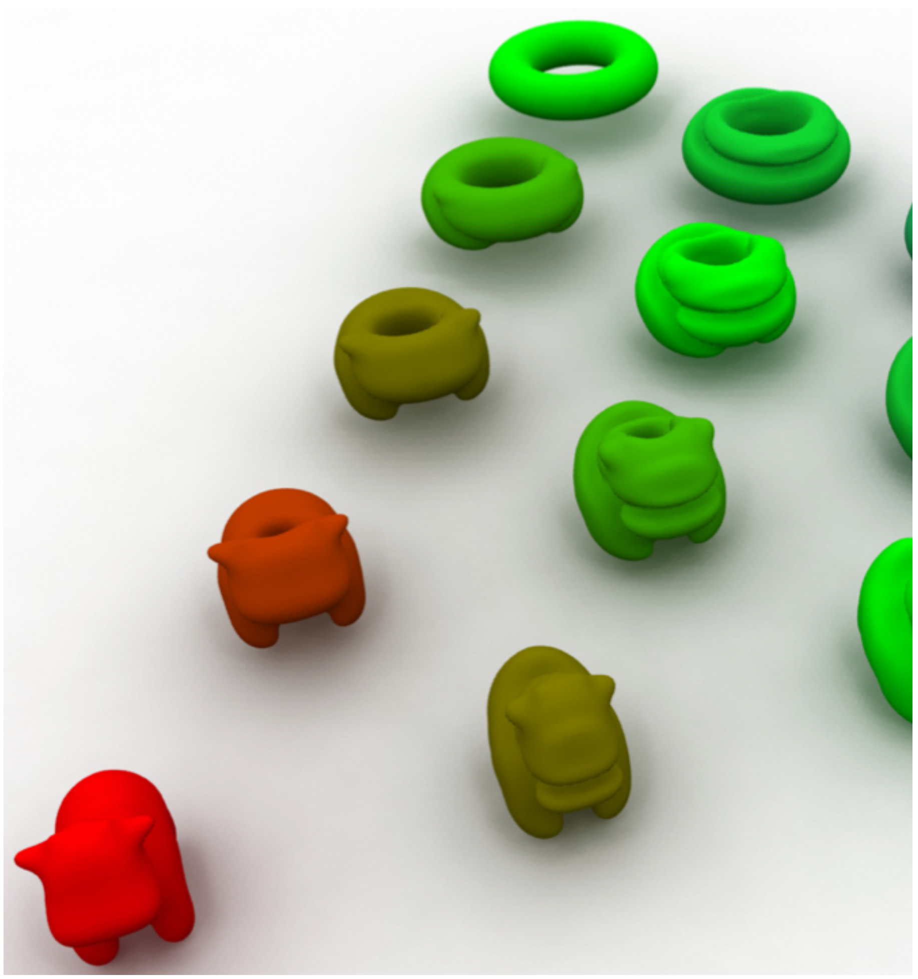
Convolutional Wasserstein Distances: Efficient Optimal Transportation on Geometric Domains,
SIGGRAPH'15 **[S..C..'15]**

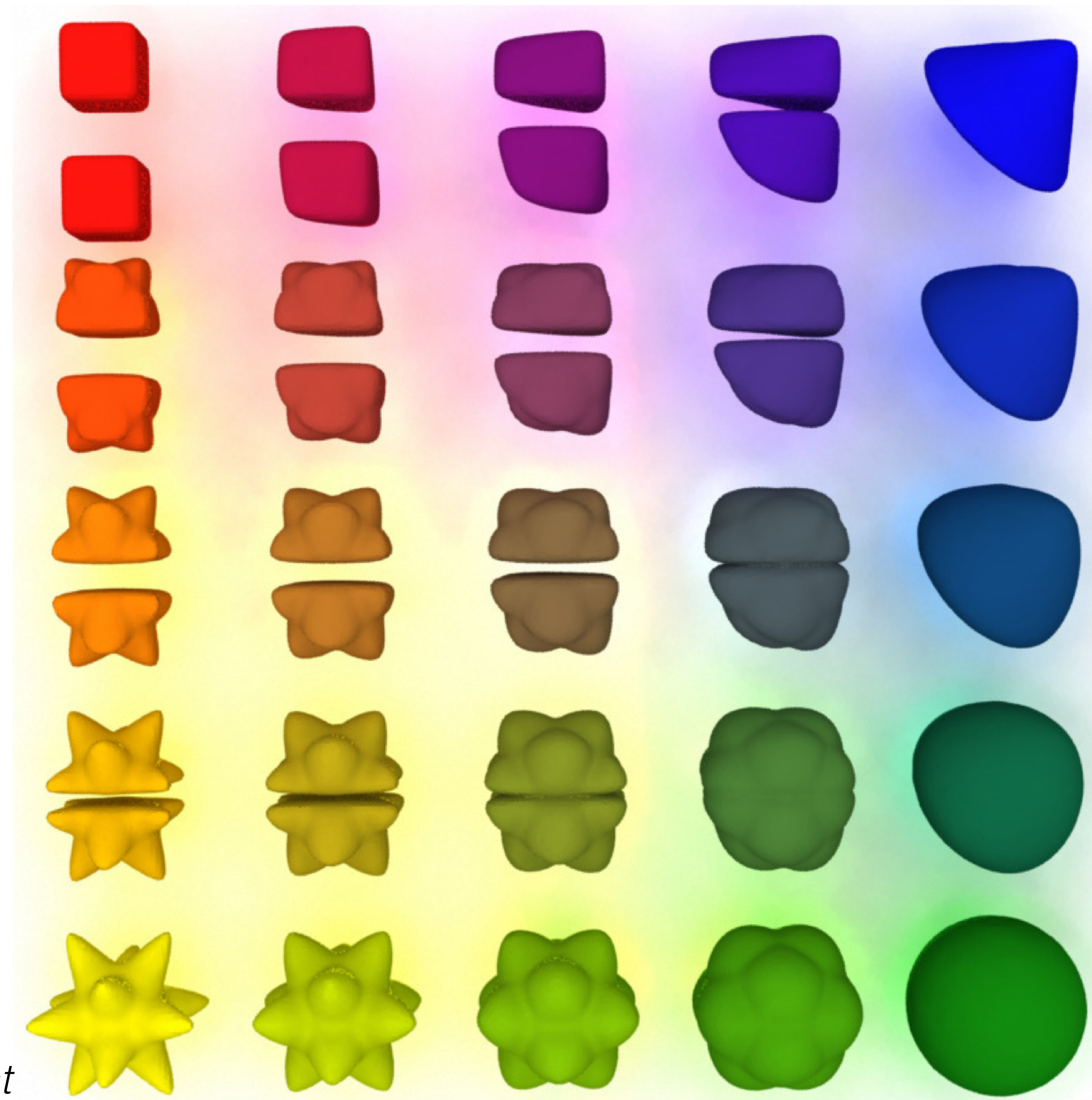
Application: Graphics



Convolutional Wasserstein Distances: Efficient Optimal Transportation on Geometric Domains,
SIGGRAPH'15 **[S..C..'15]**

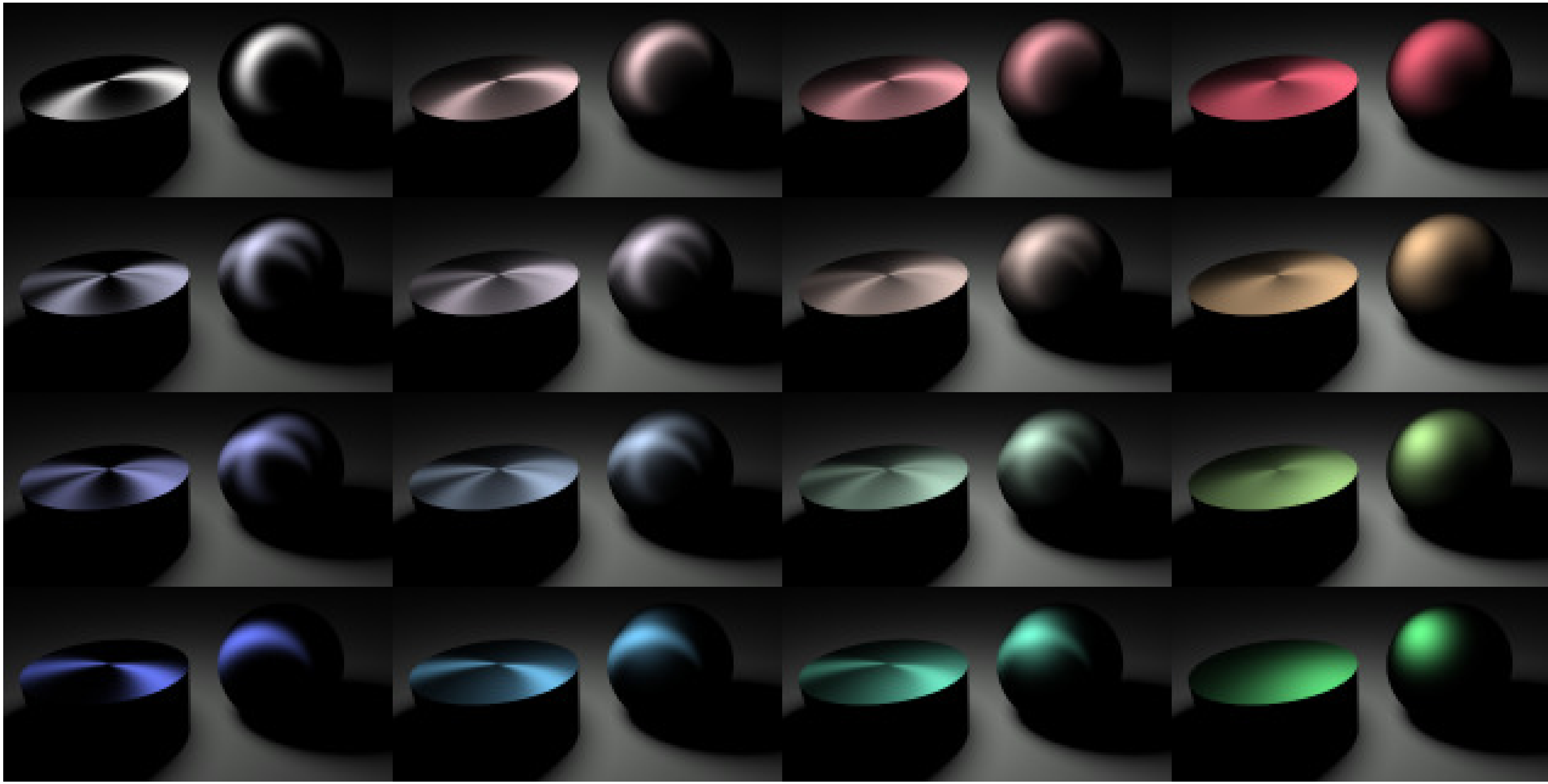
Application: Graphics





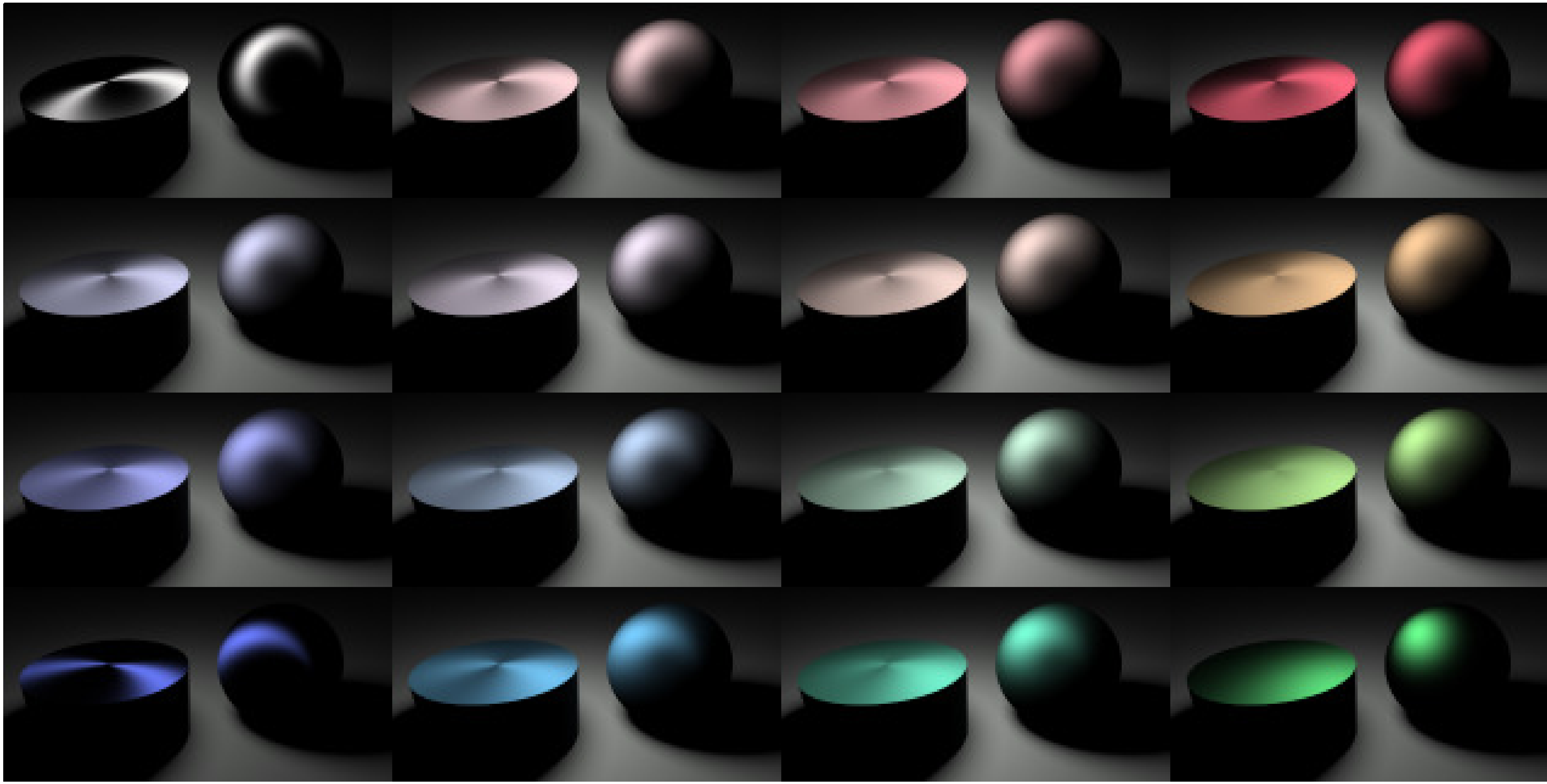
Convolutional Wasserstein Distances: Efficient Optimal Transportation on Geometric Domains,
SIGGRAPH'15 **[S..C..'15]**

Application: Graphics



Convolutional Wasserstein Distances: Efficient Optimal Transportation on Geometric Domains,
SIGGRAPH'15 **[S..C..'15]**

Application: Graphics



Convolutional Wasserstein Distances: Efficient Optimal Transportation on Geometric Domains,
SIGGRAPH'15 [S..C..'15]

Inverse Wasserstein Problems

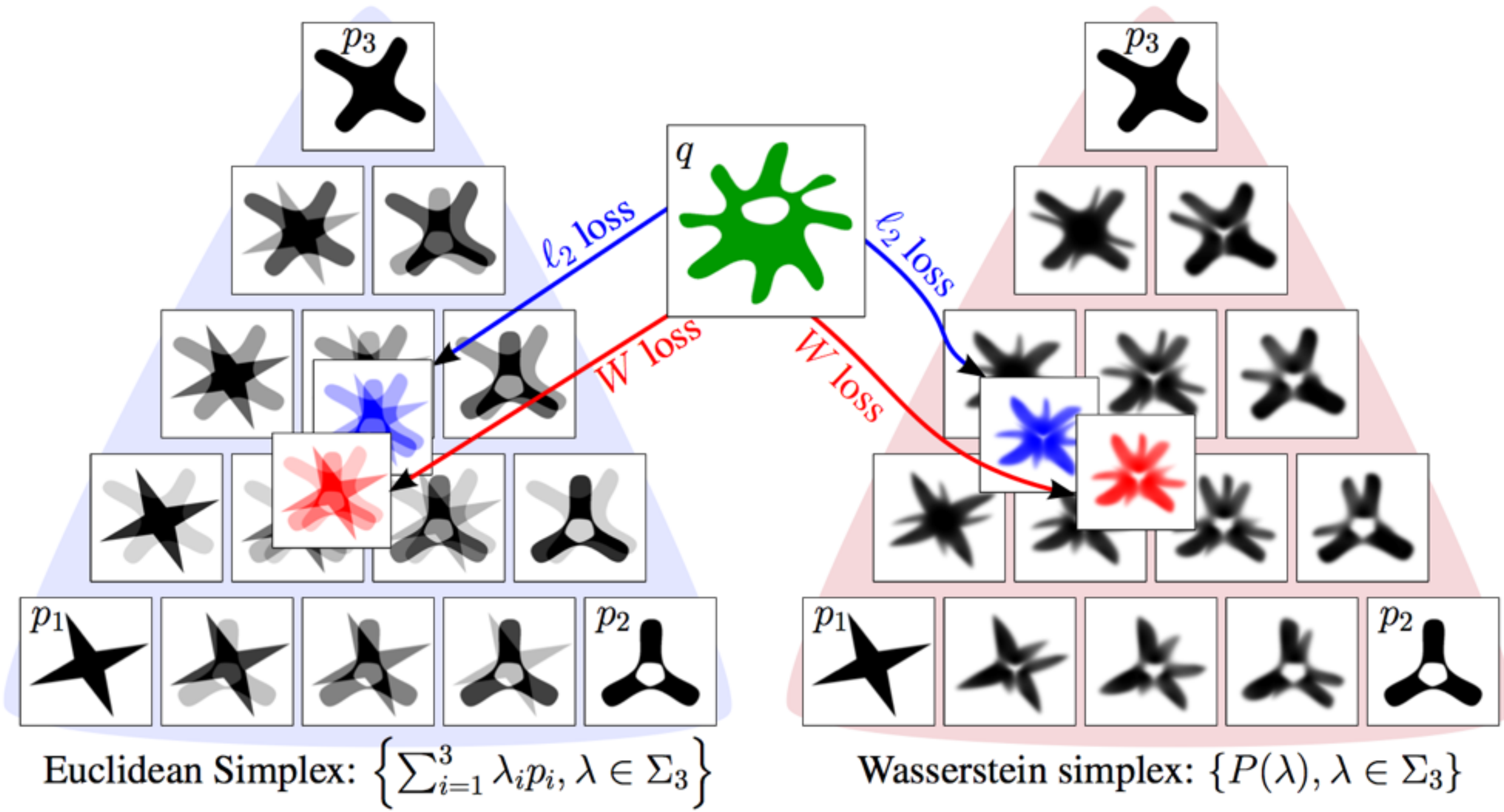
- consider Barycenter operator:

$$\boldsymbol{b}(\lambda) \overset{\text{def}}{=} \underset{\boldsymbol{a}}{\operatorname{argmin}} \sum_{i=1}^{N} \lambda_i W_\gamma(\boldsymbol{a}, \boldsymbol{b_i})$$

- address now **Wasserstein inverse problems**:

$$\text{Given } \boldsymbol{a}, \text{ find } \underset{\lambda \in \Sigma_N}{\operatorname{argmin}} \, \mathcal{E}(\lambda) \overset{\text{def}}{=} \operatorname{Loss}(\boldsymbol{a}, \boldsymbol{b}(\lambda))$$

The Wasserstein Simplex



Barycenters = Fixed Points

Prop. **[BCCNP'15]** Consider $\boldsymbol{B} \in \Sigma_d^N$ and let $\boldsymbol{U_0} = \mathbf{1}_{d \times N}$, and then for $l \geq 0$:

$$\boldsymbol{b}^l \stackrel{\text{def}}{=} \exp\left(\log\left(K^T \boldsymbol{U_l}\right)\lambda\right); \begin{cases} \boldsymbol{V_{l+1}} \stackrel{\text{def}}{=} \dfrac{\boldsymbol{b}^l \mathbf{1}_N^T}{K^T \boldsymbol{U_l}}, \\ \boldsymbol{U_{l+1}} \stackrel{\text{def}}{=} \dfrac{\boldsymbol{B}}{K \boldsymbol{V_{l+1}}}. \end{cases}$$

Using Truncated Barycenters

- instead of using the exact barycenter

$$\underset{\lambda\in\Sigma_N}{\operatorname{argmin}}\,\mathcal{E}(\lambda) \overset{\text{def}}{=} \text{Loss}(\boldsymbol{a}, \boldsymbol{b}(\lambda))$$

- use instead the L-iterate barycenter

$$\underset{\lambda\in\Sigma_N}{\operatorname{argmin}}\,\mathcal{E}^{(L)}(\lambda) \overset{\text{def}}{=} \text{Loss}(\boldsymbol{a}, \boldsymbol{b}^{(L)}(\lambda))$$

- Differente using **the chain rule.**

$$\nabla\mathcal{E}^{(L)}(\lambda) = [\partial \boldsymbol{b}^{(L)}]^T(\boldsymbol{g}),\ \ \boldsymbol{g} \overset{\text{def}}{=} \nabla\text{Loss}(\boldsymbol{a}, \cdot)|_{\boldsymbol{b}^{(L)}(\lambda)}.$$

Gradient / Barycenter Computation

function SINKHORN-DIFFERENTIATE$((p_s)_{s=1}^{S}, q, \lambda)$

$\forall s, b_s^{(0)} \leftarrow \mathbb{1}$

$(w, r) \leftarrow (0^S, 0^{S \times N})$

for $\ell = 1, 2, \ldots, L$ *// Sinkhorn loop*

$\forall s, \varphi_s^{(\ell)} \leftarrow K^\top \frac{p_s}{K b_s^{(\ell-1)}}$

$p \leftarrow \prod_s \left(\varphi_s^{(\ell)}\right)^{\lambda_s}$

$\forall s, b_s^{(\ell)} \leftarrow \frac{p}{\varphi_s^{(\ell)}}$

$g \leftarrow \nabla \mathcal{L}(p, q) \odot p$

for $\ell = L, L-1, \ldots, 1$ *// Reverse loop*

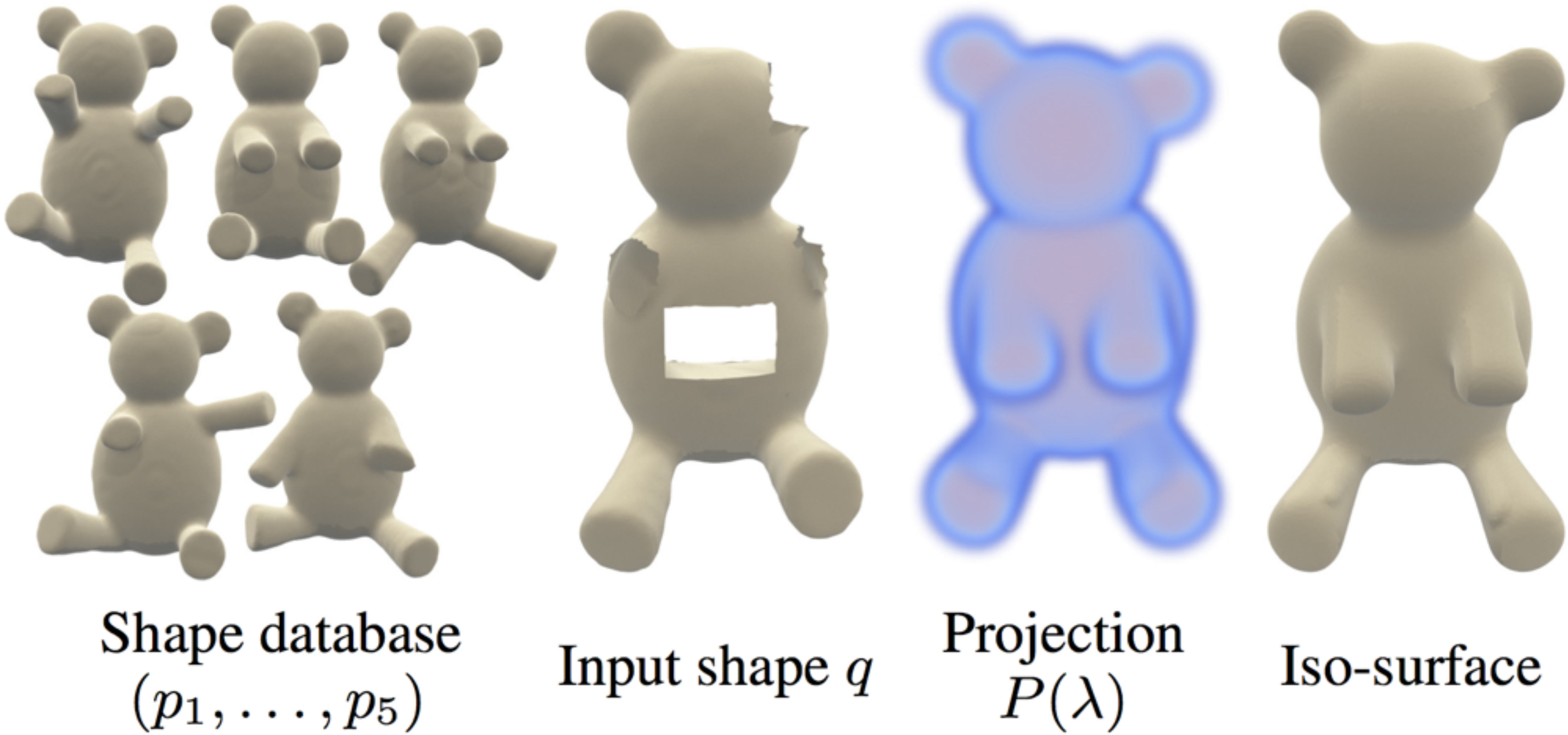
$\forall s, w_s \leftarrow w_s + \langle \log \varphi_s^{(\ell)}, g \rangle$

$\forall s, r_s \leftarrow -K^\top \left(K\left(\frac{\lambda_s g - r_s}{\varphi_s^{(\ell)}}\right) \odot \frac{p_s}{(K b_s^{(\ell-1)})^2}\right) \odot b_s^{(\ell-1)}$

$g \leftarrow \sum_s r_s$

return $P^{(L)}(\lambda) \leftarrow p, \nabla \mathcal{E}_L(\lambda) \leftarrow w$

Application: Volume Reconstruction

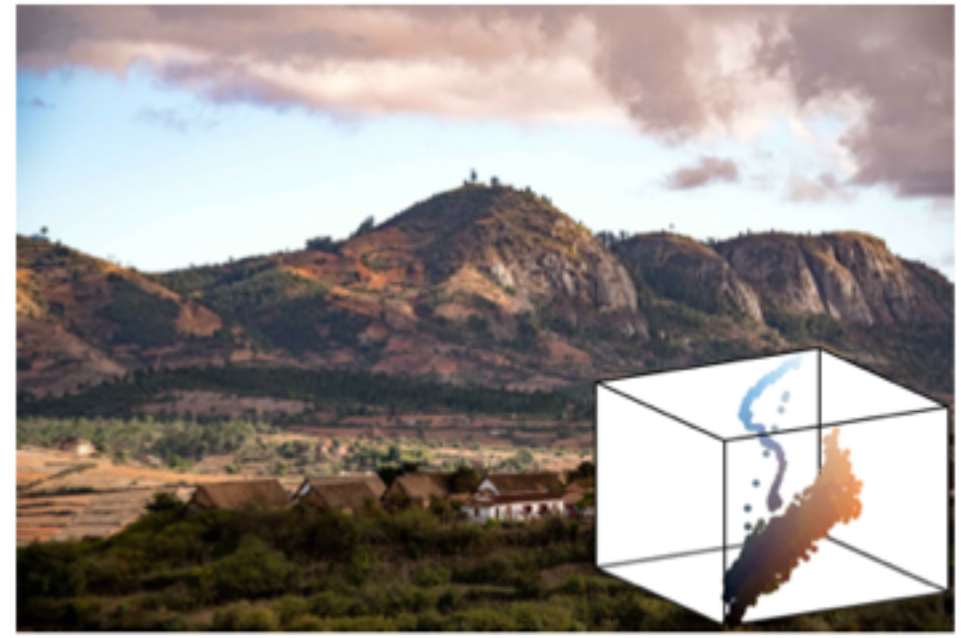


Wasserstein Barycentric Coordinates: Histogram Regression using Optimal Transport, **SIGGRAPH'16** **[BPC'16]**

Application: Color Grading

Application: Color Grading

$\lambda_0 = 0.03$



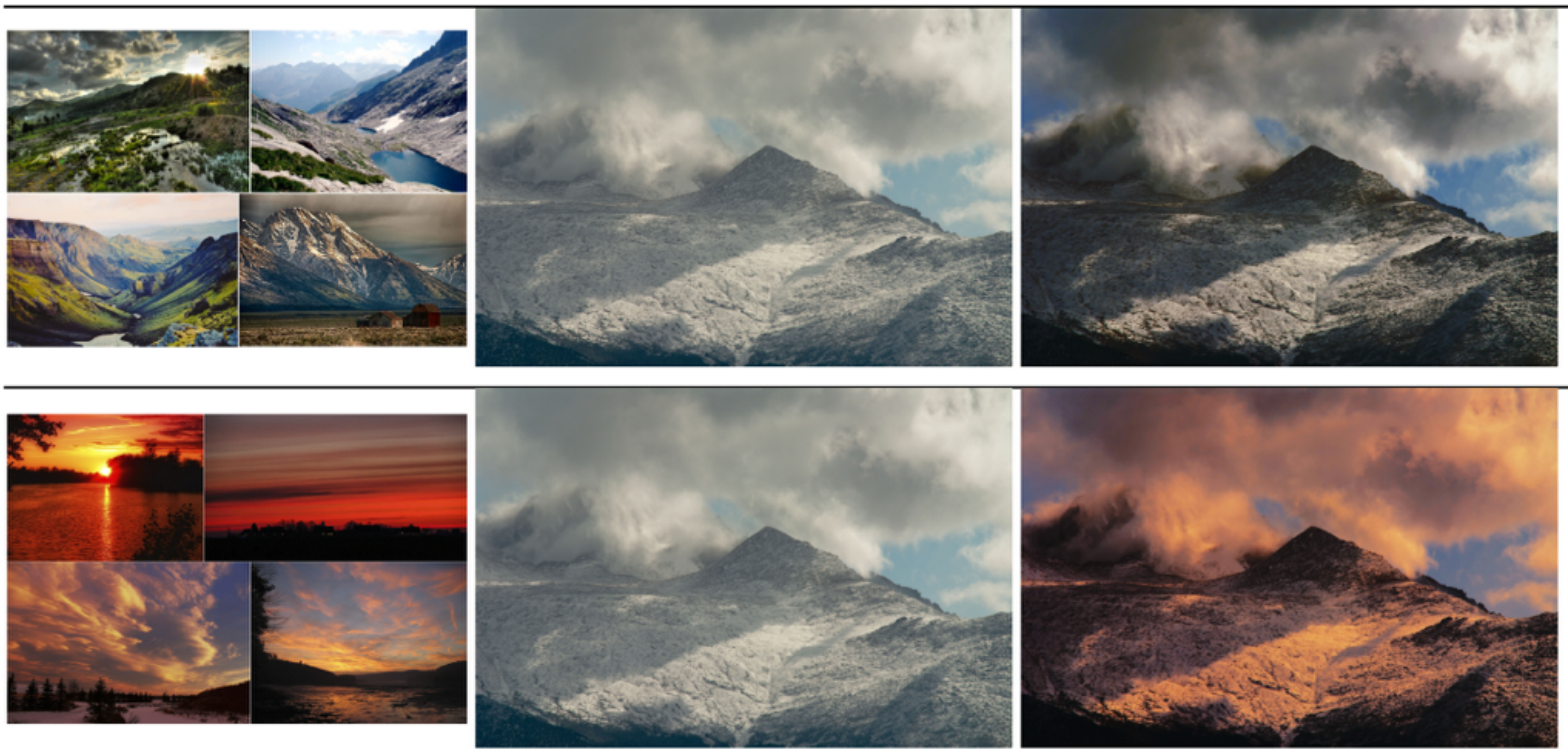
$\lambda_1 = 0.12$

$\lambda_2 = 0.40$

$\lambda_3 = 0.43$

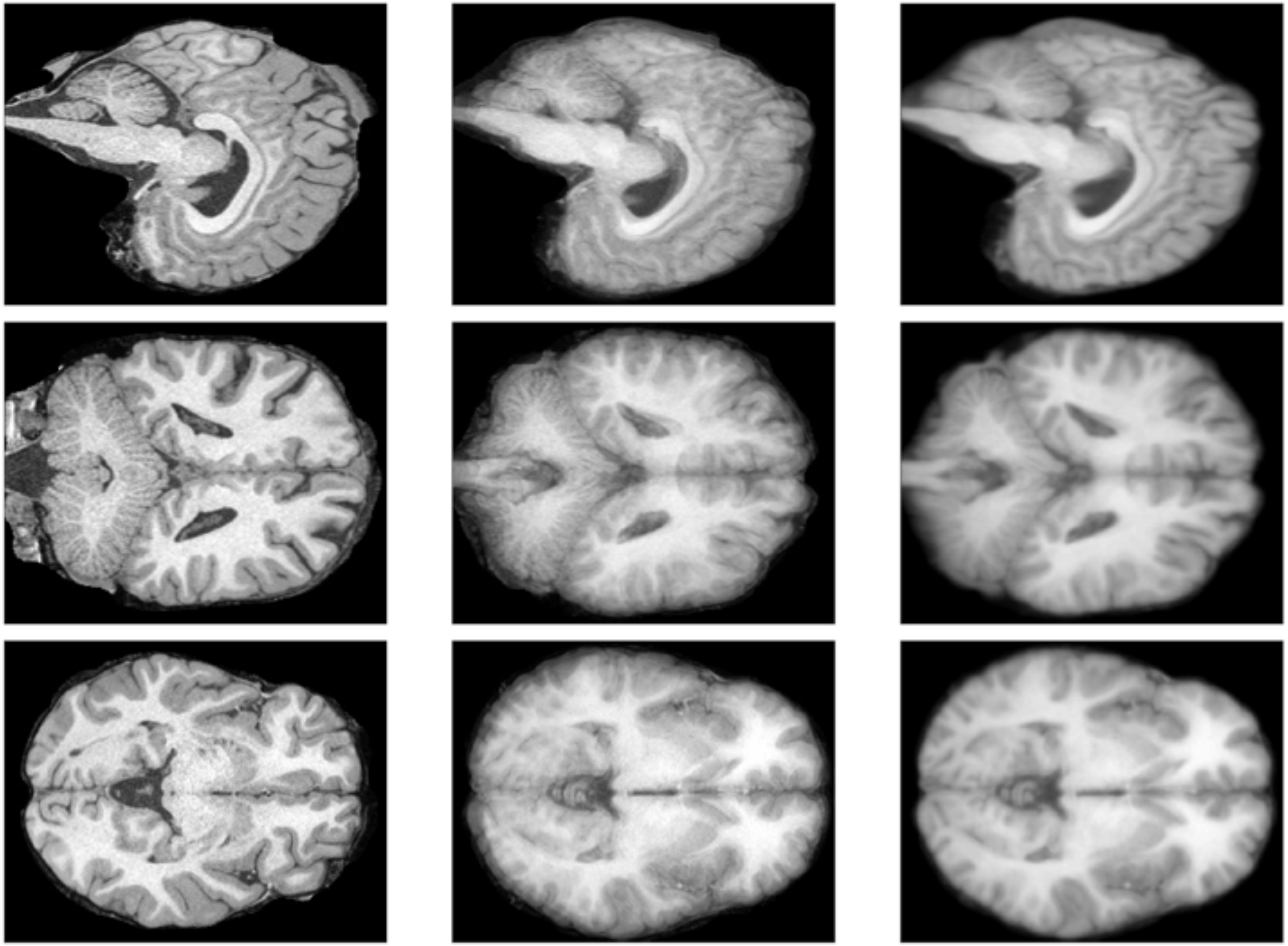
Application: Color Grading

Application: Color Grading



Wasserstein Barycentric Coordinates: Histogram Regression using Optimal Transport, **SIGGRAPH'16** **[BPC'16]**

Application: Brain Mapping



Original

Euclidean projection

Wasserstein projection

To conclude

- *Entropy* regularization is a very effective way to get OT to work as a generic loss.
- Many recent extensions:
  - **[Schmitzer'16]**: fast multiscale approaches
  - **[ZFMAP'15] [CSPV'16]**: Unbalanced transport
  - **[SPKS'16]** **[PCS'16]** extensions to *Gromov-W.*
  - **[FCTR'15]** Domain adaptation in ML