

1 几何最优传输映射

1.1 Monge-Ampère 方程

问题 1.1 (Brenier). 给定 (Ω, μ) 和 (Σ, ν) 以及成本函数 $c(x, y) = \frac{1}{2} |x - y|^2$, 最优传输映射 $T : \Omega \rightarrow \Sigma$ 是满足 Monge-Ampère 方程的 Brenier 势 $u : \Omega \rightarrow \mathbb{R}$ 的梯度映射。

$$\boxed{\det \left(\frac{\partial^2 u(x)}{\partial x_i \partial x_j} \right) = \frac{f(x)}{g \circ \nabla u(x)}} \quad (1)$$

问题 1.2 (Semi-discrete OT). 给定一个在 \mathbb{R}^d 上的紧凸域 Ω , 和 p_1, p_2, \dots, p_k 以及质量 $w_1, w_2, \dots, w_k > 0$, 找到一个最优传输映射 $T : \Omega \rightarrow \{p_1, \dots, p_k\}$, 则 $\text{vol}(T^{-1}(p_i)) = w_i$, 使运输成本最小化

$$C(T) := \frac{1}{2} \int_{\Omega} |x - T(x)|^2 dx \quad (2)$$

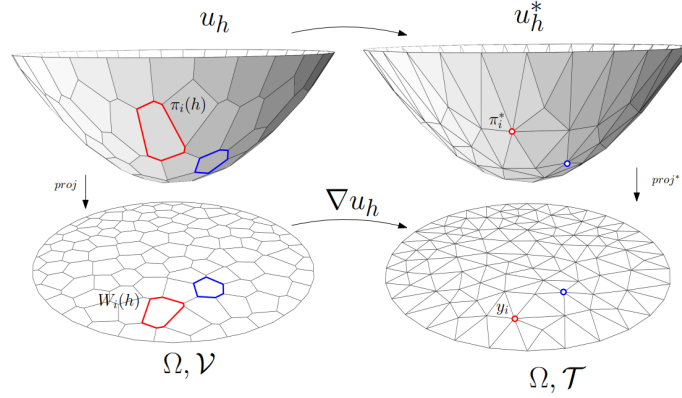


图 1: Brenier 图

根据 Brenier 定理, 将有一个分段线性凸函数 $u : \Omega \rightarrow \mathbb{R}$, 梯度映射给出了最佳运输映射。

定理 1.1 (Alexandrov 1950). 给定在 \mathbb{R}^n 上的紧凸域 Ω , 在 \mathbb{R}^n 上互不相同的 p_1, \dots, p_k , 当 $A_1, \dots, A_k > 0$, 使得 $\sum A_i = \text{Vol}(\Omega)$, 则存在 PL 凸函数

$$f(x) := \max \{ \langle x, p_i \rangle + h_i \mid i = 1, \dots, k \}, \quad (3)$$

唯一的传输使得

$$\text{Vol}(W_i) = \text{Vol}(\{x \mid \nabla f(x) = p_i\}) = A_i. \quad (4)$$

Alexandrov' s 证明是拓扑的, 而不是变分。多年来人们一直开放寻找构造性的证明。

1.2 变分证明

定理 1.2 (Gu-Luo-Sun-Yau 2013). Ω 是在 \mathbb{R}^2 上的紧凸域, y_1, \dots, y_k 在 \mathbb{R}^2 上互不相同, μ 是在 Ω 上的正连续。任意 $v_1, \dots, v_k > 0$ 以及 $\sum v_i = \mu(\Omega)$, 存在 1 个向量 (h_1, \dots, h_k) 使得,

$$u(x) = \max \{ \langle x, \mathbf{p}_i \rangle + h_i \}$$

满足 $\mu(W_i \cap \Omega) = v_i$, 其中 $W_i = \{x \mid \nabla f(x) = \mathbf{p}_i\}$ 。此外, \mathbf{h} 是凹函数的最大点

$$E(\mathbf{h}) = \sum_{i=1}^k v_i h_i - \int_0^{\mathbf{h}} \sum_{i=1}^k w_i(\eta) d\eta_i \quad (5)$$

其中 $w_i(\eta) = \mu(W_i(\eta) \cap \Omega)$ 是胞腔的 μ - 体积。

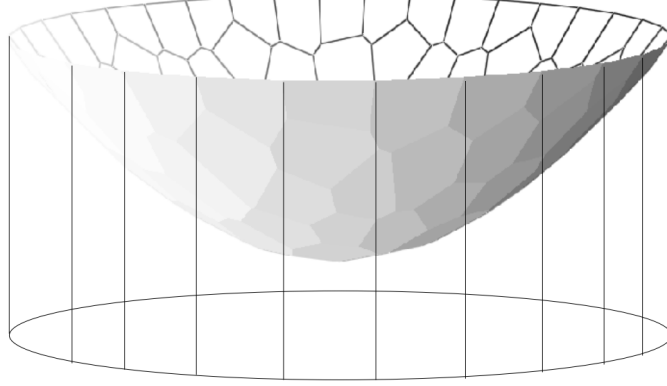


图 2: polyhedron 图

可以通过定义圆柱体 $\partial\Omega$, 圆柱体被 xy 平面和凸多面体截断。能量项 $\int^{\mathbf{h}} \sum w_i(\eta) d\eta_i$ 等于截断圆柱体的体积。

1.3 计算算法

定义 1.1 (Alexandrov Potential). 凹面能量是

$$E(h_1, h_2, \dots, h_k) = \sum_{i=1}^k v_i h_i - \int_0^{\mathbf{h}} \sum_{j=1}^k w_j(\eta) d\eta_j \quad (6)$$

能量的 Hessian 是边和双边的长度比,

$$\frac{\partial w_i}{\partial h_j} = -\frac{|\mathbf{e}_{ij}|}{[\bar{\mathbf{e}}_{ij}]}$$

Algorithm 1: Bayesian Personalized Ranking Based Latent Feature Embedding

Model

Input: latent dimension K , G , target predicate p **Output:** U^p , V^p , b^p

- 1: Given target predicate p and entire knowledge graph G , construct its bipartite subgraph, G_p
 - 2: m = number of subject entities in G_p
 - 3: n = number of object entities in G_p
 - 4: Generate a set of training samples $D_p = \{(s_p, o_p^+, o_p^-)\}$ using uniform sampling technique
 - 5: Initialize U^p as size $m \times K$ matrix with 0 mean and standard deviation 0.1
 - 6: Initialize V^p as size $n \times K$ matrix with 0 mean and standard deviation 0.1
 - 7: Initialize b^p as size $n \times 1$ column vector with 0 mean and standard deviation 0.1
 - 8: **for all** $(s_p, o_p^+, o_p^-) \in D_p$ **do**
 - 9: Update U_s^p based on Equation ??
 - 10: Update $V_{o^+}^p$ based on Equation ??
 - 11: Update $V_{o^-}^p$ based on Equation ??
 - 12: Update $b_{o^+}^p$ based on Equation ??
 - 13: Update $b_{o^-}^p$ based on Equation ??
 - 14: **end for**
 - 15: **return** U^p , V^p , b^p
-

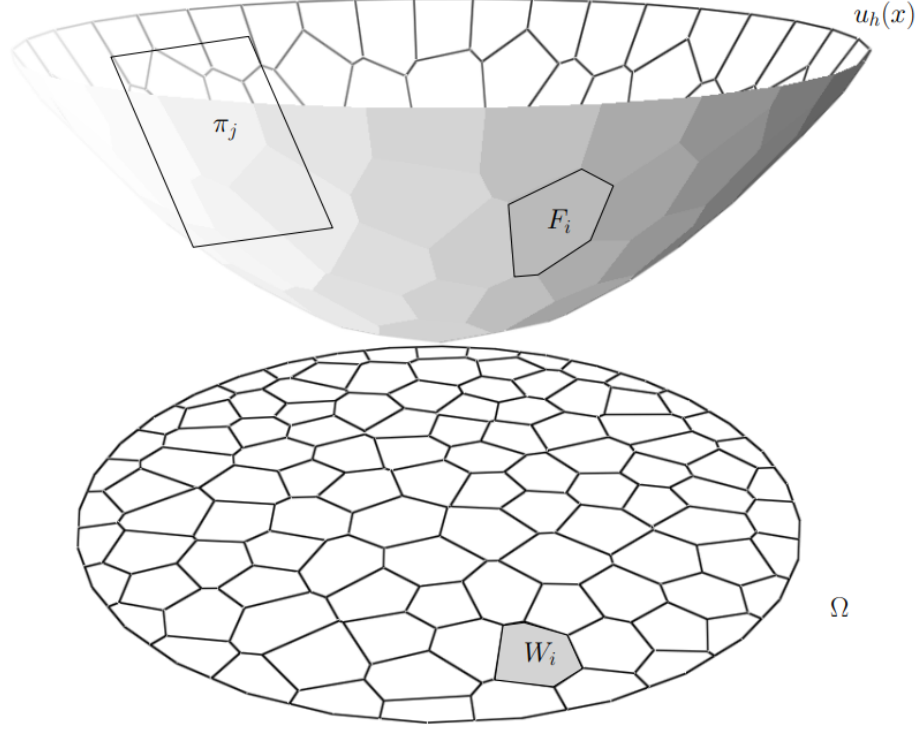


图 3: particale 图

Algorithm 2: Optimal Transport Map

Input: A set of distinct points $P = p_1, p_2, \dots, p_k$, and the weights $\{A_1, A_2, \dots, A_k\}$; A convex domain Ω , $\sum A_j = Vol(\Omega)$;

Output: The optimal transport map $T : \Omega \rightarrow P$

- 1: Scale and translate \mathbf{P} , such that $P \subset \Omega$;
- 2: Initialize $\mathbf{h}^0 \leftarrow \frac{1}{2} \left(|p_1|^2, |p_2|^2, \dots, |p_k|^2 \right)^T$;
- 3: Compute the Brenier potential $u(\mathbf{h}^k)$ (envelope of $\pi'_i s$) and its Legendre dual $u^*(\mathbf{h}^k)$ (convex hull of $\pi_i'^* s$);
- 4: Project the Brenier potential and Legendre dual to obtain weighted Delaunay trigulation $\mathcal{T}(\mathbf{h}^k)$ and power diagram $\mathcal{D}(\mathbf{h}^k)$;
- 5: Compute the gradient of the energy

$$\nabla E(\mathbf{h}) = (A_1 - w_1(\mathbf{h}), A_2 - w_2(\mathbf{h}), \dots, A_k - w_k(\mathbf{h}))^T$$

- 6: If $\|E(\mathbf{h}^k)\|$ is less than ϵ , then return $T = \nabla u(\mathbf{h}^k)$;
- 7: Compute the Hessian matrix of the energy

$$\frac{\partial w_i(\mathbf{h})}{\partial h_j} = -\frac{|e_{ij}|}{|\bar{e}_{ij}|}, \quad \frac{\partial w_i}{\partial h_j} = -\sum \frac{\partial w_i(\mathbf{h})}{\partial h_j}$$

- 8: Solve linear system

$$\nabla E(\mathbf{h}) = Hess(\mathbf{h}^k) \mathbf{d}$$

- 9: Set the step length $\lambda \leftarrow 1$;

- 10: construct the convex hull $Conv(\mathbf{h}^k + \lambda \mathbf{d})$

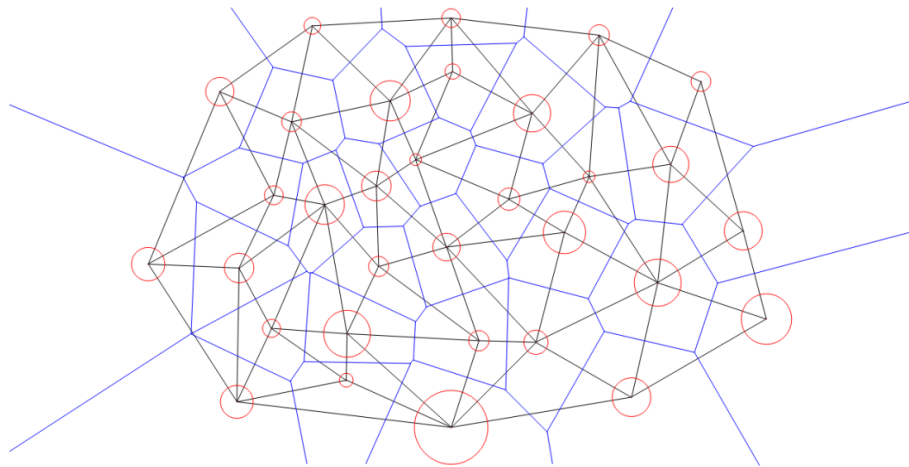


图 4: Alexandrov Potential 图

1.4 依赖关系

1. 'detri2', 一个网格生成库, 由 Si Hang 博士编写。
2. 'MeshLib', 基于半边数据结构的网格库。
3. 'freelut', OpenGL 实用工具工具包 (GLUT) 库的自由软件/开源替代品。

目录结构

- ot_2d/include, 优化传输的头文件;
- ot_2d/src, 优化传输的源文件。
- data, 一些模型。
- CMakeLists.txt, CMake 配置文件。
- resources, 需要一些资源。
- 3rdparty, MeshLib and freelut libraries.

在开始之前, 请仔细阅读自述文件, 然后一步一步地进行以下三个步骤:

1. 下载 CMake

Algorithm: ConvexHull(P)

Input: A set P of points in the plane.

Output: A list \mathcal{L} containing the vertices of $\mathcal{CH}(P)$ in clockwise order.

- 1 Sort the points by x -coordinate, resulting in a sequence p_1, \dots, p_n .
 - 2 Put the points p_1 and p_2 in a list $\mathcal{L}_{\text{upper}}$, with p_1 as the first point.
 - 3 **for** $i \leftarrow 3$ **to** n **do**
 - 4 Append p_i to $\mathcal{L}_{\text{upper}}$.
 - 5 **while** $\mathcal{L}_{\text{upper}}$ contains more than 2 points **and** the last three points in $\mathcal{L}_{\text{upper}}$ do not
 make a right turn **do**
 - 6 Delete the middle of the last three points from $\mathcal{L}_{\text{upper}}$.
 - 7 **end**
 - 8 **end**
 - 9 Put the points p_n and p_{n-1} in a list $\mathcal{L}_{\text{lower}}$, with p_n as the first point.
 - 10 **for** $i \leftarrow n - 2$ **downto** 1 **do**
 - 11 Append p_i to $\mathcal{L}_{\text{lower}}$.
 - 12 **while** $\mathcal{L}_{\text{lower}}$ contains more than 2 points **and** the last three points in $\mathcal{L}_{\text{lower}}$ do not
 make a right turn **do**
 - 13 Delete the middle of the last three points from $\mathcal{L}_{\text{lower}}$.
 - 14 **end**
 - 15 **end**
 - 16 Remove the first and the last point from $\mathcal{L}_{\text{lower}}$ to avoid duplication of the points
 where the upper and lower hull meet.
 - 17 Append $\mathcal{L}_{\text{lower}}$ to $\mathcal{L}_{\text{upper}}$, and call the resulting list \mathcal{L} .
 - 18 **return** \mathcal{L}
-