Optimal transport and the pancake cut

Johanna Siemelink

May 2024

Introduction

This semester my goal was to familiarize myself with the optimal transport problem and write a program that solves the discrete version between two sets of points of equal size. The hope is that increasing the number of points to approximate a continuous set may give insight to the general optimal transport of simple shapes in two dimensions.

I followed an open access online course [1] to learn about the optimal transport problem, and read up on the literature relevant to the goal. I wrote a simple program testing the waters with different algorithms I found. I also implemented an algorithm that computes the pancake cut. This report is an introduction to optimal transport to demonstrate some of what I have learnt, and to outline the mathematical background of the algorithms I implemented.

The optimal transport problem

The question of optimal transport arises any time something has to be moved from one place to another. The problem was first studied by Gaspard Monge, a French mathematician from the 18th century. Legend has it that the question arose to help Napoleon build defences efficiently: how best to move dirt from a moat to form a wall. This is the mathematical question Monge proposed: When given two density functions μ on X and ν on Y how can we minimize a cost function c when moving everything from μ to ν.

Monge formulation

First let us define what moving a measure into another one means. To have a chance at mass balance we need $\mu(\mathbb{R}^n) = \nu(\mathbb{R}^n)$. We want the transport to go from supp $\mu = X$ to supp $\nu = Y$, let's call it a transport map $T : X \to Y$. For every set to retain mass we need:

$$\mu(T^{-1}(A)) = \nu(A) \qquad \forall A \subseteq Y$$

The lefthand side of this equation is called the pushforward of T, its notation is $T_{\#}\mu$

And of course there is a cost function $c : X \times Y \to \mathbb{R}$. When thinking about the one dimensional optimal transport problem it is quickly apparent that there can be multiple optimal solutions if we only use distance as the cost function. This is why, even for simple questions, the quadratic cost function is often used. All in all, this is the Monge formulation:

$$minimise\left\{ \int_{\mathbb{R}} c(x, T(x)) d\mu(x) \middle| T_{\#}\mu = \nu \right\}$$

Kantorovich formulation

When looking at the following discrete measures, a problem arises with Monge's formulation. Let μ be a single point, with mass 2, and ν be two points both with mass 1. This problem is not feasible, even though anyone could easily devise a way to transport the mass from μ to ν. This means we need to redefine everything mathematically to be able to send half the flour from the mill to one bakery and half to the other, so to speak.

We need what we will call a transport plan that tells us how much mass is moved from one set in X to another one in Y. Of course it has to be non negative: $\Pi : X \times Y \to \mathbb{R}^+$. This is not enough, we want it to be mass preserving. Let us see how that looks for an arbitrary set $A \subseteq X$ we want the total value of its image to be $\mu(A)$ so we'd like $\Pi(A, Y) = \mu(A)$. Symmetrically we want the inverse image $\Pi(X, B) = \nu(B)$ for all $B \subseteq Y$. The set of transport maps that fulfill the marginal constraints are denoted: $\Pi(\mu, \nu)$. And this is enough, so the question becomes:

$$minimise\left\{ \int_{X\times Y} c(x, y) d\Pi(x, y) \middle| \Pi \in \Pi(\mu, \nu) \right\}$$

Discrete Kantorovich

In the discrete setting our goal is to min $\sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij}\Pi_{ij}$ where Π is our transport plan. Π can't transport negative mass from one point to another, so we want $\Pi_{ij} \geq 0$. The marginals in this case give us constrains with which we can now state the discrete transport problem as an LP to solve for Π:

Constraints:

$$\sum_{i=1}^{n} \Pi_{ij} = \mu_j, \; \sum_{j=1}^{m} \Pi_{ij} = \nu_i, \; \Pi_{ij} \geq 0$$

Objective function:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij}\Pi_{ij}$$

Linear programs have a dual optimization problem, this one does as well, it can be very useful. It even has a generalization for the non-discrete case, often used to check if a given transport plan is optimal.

Entropic regularization

Now our goal won't be to find an optimal solution, but one close enough. If our linear program was $\mathrm{L}(\mu, \nu) = min_{\Pi \in \Pi(\mu,\nu)} < c, \Pi >$ our modified linear program will be using entropy H as follows:

$$\mathrm{L}^{\epsilon}(\mu, \nu) = min_{\Pi \in \Pi(\mu,\nu)} \left\{ < c, \Pi > -\epsilon \mathrm{H}(\Pi) \right\} \qquad \mathrm{H} = -\sum_{i=1}^{n} \sum_{j=1}^{m} \Pi_{ij} (\log \Pi_{ij}) - 1$$

Thankfully it can be shown that $\lim_{\epsilon \to \infty} \mathrm{L}^{\epsilon}(\mu, \nu) = L(\mu, \nu)$, so this relaxation makes sense! We will use this concept later to apply Sinkhorn's alorithm in this setting.

Numeric methods

To compute a continuous problem, a good idea is to take the discrete version and try to approximate the continuous version with it. I tried out two of these:

Hungarian method

If we take the non-intersecting discrete optimal transport in the Monge setting, our problem becomes to move each point from one set to one in the other set. This means the objective is a matching, with minimal c weight. The Hungarian method is a graph algorithm that returns a minimal weight matching in a bipartite graph. For more details see [2] I did however use it in my program as a baseline to compare how close algorithms that solve a relaxed version come to the optimum, and also to compare it to the pancake conjecture (see below).

Sinkhorn's method

In Sinkhorn 1967 [3] the original problem was to rescale the rows and columns of a matrix to make it doubly stochastic, which means that every row every column sums to 1, in other words its marginals are $\mathbb{1}_n$ and $\mathbb{1}_m$. To use this for optimal transport we look at Π as a $\mathbb{R}^{n \times m}$ matrix, where we want the marginals to be μ and ν. So the idea [4] is to calculate Π^{ϵ} with Sinkhorn's algorithm. Take Π^{ϵ} as $\mathrm{diag}(u) K \mathrm{diag}(v)$, it turns out the right K to use is $K_{ij} = e^{\frac{c_{ij}}{\epsilon}}$. Rearranging the marginal constraints gives us $u \odot (Kv) = \mu$ and $(Ku) \odot v = \nu$. Modifying Sinkhorn's algorithm to iteratively attain these marginals the steps become:

$$u^{k+1} = \mu \oslash (K v^{k})$$

$$v^{k+1} = \nu \oslash (K u^{k+1})$$

This algorithm is very fast, but it loses accuracy by solving only the entropically regularized problem. But with a smaller and smaller ϵ the approximation should get better and better! Unfortunately a very small ϵ creates a very small $e^{\frac{c_{ij}}{\epsilon}}$, which quickly became too small to handle in my program, leaving me with only approximate matchings, though they are very close to optimal in the cases I checked.

Pancake conjecture

My thesis supervisor proposed the idea to investigate if there are any links between the pancake cut and the optimal transport in 2 dimensions. The pancake cut is the two dimensional version of the ham and sandwich problem. In the ham and sandwich problem two pieces of bread and a ham are somewhere in a 3d space. The goal is to share them equally for two: two equal pieces of top bun, bottom bun and ham. The famous theorem states that the sandwich can always be sliced into two equal parts with one plane. The two dimensional version is sometimes referred to as the pancake problem: slice two pieces of pancake in half with one line.

Conjecture: There exists an optimal transport map where the disected halves of the pancake transfer to the other pancake's disected halves without mixing. If this were true for all pancakes of any form, this could be applied recursively to the two halves, then the two quarters, then the eights and so forth, giving us the transfer map in some form.

Megiddo's algorithm

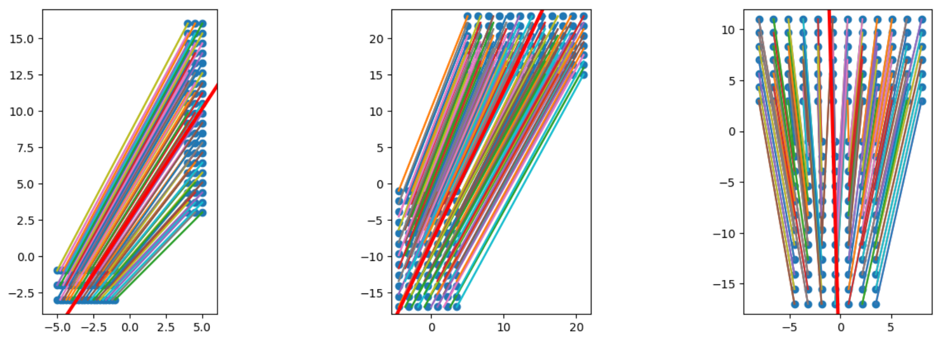
To combine this with our previous approximation I coded Megiddo's algorithm [5], which solves the discrete pancake problem.

Given are n points (a_i, b_i) where $\forall b_i \geq 0$ and m points (c_i, d_i) where $\forall d_i \leq 0$. If we take their point-line duals, often used in computational geometry, we get two sets of lines: $y = \frac{a_i}{bi} + \frac{-1}{b_i}x$ and $y = \frac{c_i}{di} + \frac{-1}{d_i}x$. The dual of the shared median line becomes the shared median point, that is the intersection of the pointwise median functions: $P_m(x) = median(\frac{a_i}{bi} + \frac{-1}{b_i}x | \forall i = 1, ..., n)$ and $Q_m(x) = median(\frac{c_i}{di} + \frac{-1}{d_i}x | \forall j = 1, ..., m)$. So the point we're looking for is the x^* where $Q_m(x) = P_m(x) =: y^*$, let's call it (x^*, y^*).

At the heart of this algorithm lies a line search query which determines on which side of the line (x^*, y^*) is. Megiddo's program is the first prune and search program: it takes the lines and with two smartly placed line queries it finds at least 1/8 of lines that can be discarded. It then iteratively repeats this until we get the right answer.

Goals

This semester I have written a program that builds an optimal transport map and a pancake cut. The results look promising based on the images created, but to properly compare these to each other we need some type of evaluation. This is the initial plan for the next project, further plans depend on those results.



The pancake cut and an optimal matching in three layouts

References

[1] Hamfeldt, Brittany (2019): Introduction to Optimal Transport. – New Jersey Institute of Technology
https://www.youtube.com/playlist?list=PLJ6garKOlK2qKVhRm6UwvcQ46wK-ciHbl

[2] Frank, András (revised 2010): Connections in Combinatorical Optimization. 3.3.1 The Hungarian method

[3] Sinkhorn, Richard, & Knopp, Paul (1967): Concerning nonnegative matrices and doubly stochastic matrices. – Pacific J. Math. 21, 343–348.

[4] Cuturi, Marco (2013): Sinkhorn distances: Lightspeed computation of optimal transport. – Advances in neural information processing systems. 2292–2300.

[5] Megiddo, Nimrod (1985): Partitioning with two Lines in the Plane. – Journal of algorithms 6. 430-433

[6] Zhang, Hengning: ”Megiddo’s O(n) Linear Programming Visualization”
https://hengningzhang.com/posts/megiddo/