Fermat-Torricelli Problem with Weights

Hon Leung Lee

June 10, 2013

1 Introduction

This is a popular problem in location science: Given $m \geq 2$ distinct points $a_1, \cdots, a_m$ in $\mathbb{R}^d$ and positive weights $\eta_1, \cdots, \eta_m$. Define $F : \mathbb{R}^d \to [0, \infty)$ given by $F(x) = \sum_{i=1}^{m} \eta_i \|x - a_i\|$ where $\|\cdot\|$ is the standard Euclidean norm. Fermat-Torricelli Problem is the minimization of $F(x)$ over $\mathbb{R}^d$. The minimizer of this problem is called the Fermat-Torricelli point. The goal is to use `Python` / `Sage` [4] to compute such point.

If $a_1, \cdots, a_m$ are non-collinear, then minimizer exists and is unique, because of the strict convexity of F. Otherwise, solution exists but may not be unique. For simplicity we always assume the non-collinearity.

In the case $d = 2$, $m = 3$ and $\eta_1 = \eta_2 = \eta_3 = 1$, the problem was proposed by Pierre de Fermat in the early 17th century and solved by Evangelista Torricelli. This justifies the name of the problem. We refer readers to [2] and the references therein for the extensions and applications of this problem.

We shall solve this problem using three methods and then compare their performance. The first method is called modified Weiszfeld algorithm described in [5]. The second method involves formulating the problem as a second order cone programming (SOCP) problem. Thirdly the problem with $d = 2$ can be solved by formulating it as a semidefinite programming (SDP) problem [3].

The `cvxopt` package in `Python` is the main tool for us to handle this problem.

2 Modified Weiszfeld Algorithm

The idea of the original Weiszfeld algorithm is to construct a minimizing sequence converging to the solution, if the initial point is chosen correctly. Otherwise the algorithm gets stuck at one of the vertices $a_1, \cdots, a_n$ which may not actually be the solution. The modified algorithm solves this problem and thus improves the original version.

Definition 1. *Given non-collinear points $\{a_1, \cdots, a_m\} \subset \mathbb{R}^d$ and positive weights $(\eta_1, \cdots, \eta_m)$ for $m \geq 2$. Define $\beta : \mathbb{R}^d \to \mathbb{R}$, $\omega_i : \mathbb{R}^d \to \mathbb{R}$ (for $i = 1, \cdots, m$) and $\widetilde{T} : \mathbb{R}^d \to \mathbb{R}^d$ as follows:*

$$\beta(x) = \left(\sum_{a_j \neq x} \frac{\eta_j}{\|x - a_j\|} \right)^{-1}, \quad \omega_i(x) = \frac{\eta_i \beta(x)}{\|x - a_i\|} \text{ and } \widetilde{T}(x) = \sum_{a_i \neq x} \omega_i(x) a_i.$$

Also define $\eta : \mathbb{R}^d \to \mathbb{R}_+$ by $\eta(x) = \eta_k$ if $x = a_k$, $k = 1, \cdots, m$, or 0 otherwise. Plus define $r : \mathbb{R}^d \to \mathbb{R}_+$ by $r(x) = \|\widetilde{R}(x)\|$ where $\widetilde{R}(x) \triangleq \sum_{a_i \neq x} \frac{\eta_i (a_i - x)}{\|a_i - x\|}$.

The modified algorithm indeed computes a fixed point of $T : \mathbb{R}^d \to \mathbb{R}^d$ given by

$$T(x) = \begin{cases} \left(1 - \dfrac{\eta(x)}{r(x)}\right)^+ \widetilde{T}(x) + \min\left\{1, \dfrac{\eta(x)}{r(x)}\right\} x & \text{if } r(x) > 0 \\ x & \text{if } \eta(x) > r(x) = 0 \\ \widetilde{T}(x) & \text{if } \eta(x) = r(x) = 0 \end{cases}$$

The coming theorem was proved in [5].

Theorem 2 (Modified Weiezfeld algorithm)**.** *Given any $x^{(0)} \in \mathbb{R}^d$ and define $x^{(n)} = T(x^{(n-1)})$ for all $n \geq 1$. Then:*

(1) $\{x^n\}$ converges to the unique minimizer x^* of the Fermat-Torricelli problem.

(2) If $r(x^{(k)}) = 0$ or $\eta(x^{(k)}) \geq r(x^{(k)}) > 0$ for some k, then $x^{(n)} = x^*$ for all $n \geq k$.

Therefore the algorithm can be implemented by taking $r(x^{(n)}) = 0$ or $\eta(x^{(n)}) \geq r(x^{(n)}) > 0$, together with $n \leq$ some fixed big number say 100, as the stopping criteria. This algorithm shows the possibility of solving the problem with Euclidean norm replaced by l^p norm in $\mathbb{R}^d$.

3 Second Order Cone Programming

Our problem can be formulated as a Second Order Cone Programming (SOCP) problem and then solved using the solver in the package `cvxopt`.

It is immediate that the given problem is equivalent to this SOCP:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{m} \eta_i d_i \\ \text{subject to} \quad & d_i \geq \|x - a_i\| \text{ for all } i \\ & x \in \mathbb{R}^d, d_i \in \mathbb{R} \text{ for any } i \end{aligned}$$

Indeed any SOCP can be formulated as a semidefinite programming (SDP) using the relation:

$$\|u\| \leq t \text{ if and only if } \begin{bmatrix} tI & u \\ u^T & t \end{bmatrix} \succeq 0.$$

However it is known (see [1]) that solving SOCP directly has a much better worse case complexity than such SDP approach. Notice this SDP approach is different from the one in the next section.

We remark that when running the SOCP solver in `cvxopt`, the inputs should be dense vectors. If the inputs are too sparse the solver may not work. This is because the solver requires matrix inverse computation. Sparse inputs may increase the chance of singularity of matrices.

4 Semidefinite Programming

We describe here the idea of SDP formulation in [3]. Our problem ($d = 2$ case) is to minimize over $x \in \mathbb{R}^2$, $F(x) = \sum_{i=1}^{m} \eta_i \|x - a_i\|$ where all η_i is positive and $a_i \in \mathbb{R}^2$. Let x and a_i have coordinates

(x, y) and (a_i, b_i) respectively (some abuse of notation here). Then the objective function becomes

$$F(x, y) = \sum_{i=1}^{m} \eta_i \sqrt{(x - a_i)^2 + (y - b_i)^2}.$$

Let the minimization problem be $(\mathcal{P})$.

Introduce slack variables $d_i = \sqrt{(x - a_i)^2 + (y - b_i)^2}$ for all i. Then $d_i^2 = (x - a_i)^2 + (y - b_i)^2$. One can use this to show that:

Theorem 3. *The problem $(\mathcal{P})$ is equivalent to the semidefinite program (SDP) $(\mathcal{P}_2)$:*

$$\textit{minimize } F(x) = \sum_{i=1}^{m} \eta_i d_i \textit{ subject to } \quad \begin{bmatrix} d_i + x - a_i & y - b_i \\ y - b_i & d_i - x + a_i \end{bmatrix} \succeq 0 \textit{ for all } i = 1, \cdots, n$$
$$d_1, \cdots, d_m, x, y \in \mathbb{R}$$

(Here $A \succeq 0$ means the matrix A is positive semidefinite).

Proof. For each j let $A_i = \begin{bmatrix} d_i + x - a_i & y - b_i \\ y - b_i & d_i - x + a_i \end{bmatrix}$. If $d_i = \sqrt{(x - a_i)^2 + (y - b_i)^2}$ then all principal minors of A_i are nonnegative: $d_i \geq |x - a_i|$ and $\det(A) = d_i^2 - (x - a_i)^2 - (y - b_i)^2 = 0$. Thus $A_j \succeq 0$. So feasibility of $(\mathcal{P})$ implies the feasibility of $(\mathcal{P}_2)$, hence the minimum value in $(\mathcal{P}_2)$ cannot exceed that of $(\mathcal{P})$. On the other hand if $A_i \succeq 0$, then all the eigenvalues of A_i are nonnegative. By assessing the eigenvalues using quadratic formula one sees $d_i \geq \sqrt{(x - a_i)^2 + (y - b_i)^2} \geq$ the minimum value of $(\mathcal{P})$. This completes the other direction that the minimum value of $(\mathcal{P})$ is at most that of $(\mathcal{P}_2)$, and so the conclusion follows. □

The above SDP $(\mathcal{P}_2)$ can be implemented using the SDP solver in `cvxopt` package in Python. So far this method only works for $d = 2$. Possible future work is how to generalize this idea to high dimensions.

5 Implementation

For the details of implementation please refer to the `Sage` worksheet or the `Python` codes attached.

6 Which algorithm is the best?

We test these algorithms using the following way: Set $d = 2$. Construct 100 data sets which each data set has 4 given points. Their coordinates are independent identically distributed as $N(0, 10)$ (normal distribution with mean 0 and standard deviation 10). For each data set we run all three algorithms, then compare the median number of iterations plus the median CPU time. This is the result:

Modified Weiszfeld algorithm median number of iterations: 100.0
Modified Weiszfeld algorithm median time: 0.00225400924683
SOCP median number of iterations: 7.0
SOCP median time: 0.00238013267517
SDP median number of iterations: 7.0
SDP median time: 0.00225901603699

We remark that the above test cannot be performed in SageMathCloud [4] since the output is automatically truncated. However it can be performed in Sage Notebook [4].

The advantage and drawback of each algorithm is summarized here.

	Advantage	Drawback
Modified Weiszfeld	Works in d dimensions Possibility to solve the corresponding l^p norm problem	Slow, in terms of iteration number
SOCP	Works in d dimensions Fast, in terms of iteration number	Inputs must be dense
SDP	Fast, in terms of iteration number	Works only in $d = 2$

Thus there is no absolutely the best algorithm.

References

[1] Lobo, M. S., Vandenberghe, L., Boyd, S., Lebret, H. (1998). *Applications of second-order cone programming.* Linear algebra and its applications, 284(1), 193-228.

[2] Mordukhovich, B.S., Nam, N.M. (2011). *Applications of variational analysis to a generalized Fermat-Torricelli problem.* J. Optim. Theory Appl., 148, 431-454.

[3] Nie, J., Parrilo, P.A. Sturmfels, B. (2008). *Semidefinite Representation of the k-Ellipse. In Algorithms in Algebraic Geometry.* Springer New York, 117-132.

[4] Stein, W. et al (2011). *Sage Mathematics Software (Version 4.7).* The Sage Development Team, `http://www.sagemath.org`. (Sage Notebook Version 5.4, `http://uw.sagenb.org/`; SageMathCloud, `https://cloud.sagemath.com/`)

[5] Vardi, Y., Zhang, C-H. (2001). *A modified Weiszfeld algorithm for the Fermat-Weber location problem.* Math. Program., Ser. A 90, 559-566.