# Optimization project

Thomas Torkildsen, Øyvind Hansen Singsaas, Simon Liabø and Magnus Solheim Grytten

April 2021

## 1 Definitions

### 1.1 Convex function in $\mathbf{R}^n$

A function is convex if $\forall x, y \in \mathbf{R}^n$ and $0 \leq \lambda \leq 1$. Then

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) \tag{1}$$

holds.

### 1.2 Properties of convex functions

$$\text{If } f, g\text{: } \mathbb{R}^n \to \mathbb{R} \text{ is convex, then } f+g \text{ is convex.} \tag{2}$$

$$\text{if } f, g\text{: } \mathbb{R}^n \to \mathbb{R} \text{ is convex, then } h(x) = max\{f(x), g(x)\} \text{ is convex.} \tag{3}$$

### 1.3 Minkowski's Inequality for Sums

Let $a_1$, ..., $a_n$, $b_1$, ..., $b_n \in \mathbb{R}_{\geq 0}$ and let $p > 1$, then:

$$(\sum_{i=1}^{n}(a_i + b_i)^p)^{\frac{1}{p}} \leq (\sum_{i=1}^{n}(a_i)^p)^{\frac{1}{p}} + (\sum_{i=1}^{n}(b_i)^p)^{\frac{1}{p}} \tag{4}$$

## 2 Project Tasks - Theoretical Work

### 2.1 Task 1

By the definition of a metric in the problem description we show that all three conditions hold for $d_1$, $d_2$ and $d_\infty$, and thus are metrics.

Showing that $d_1$ is a metric.

1. condition: $d_1(x, y) = 0 \implies x = y$.

$$d_1(x, y) = |x_1 - y_1| + |x_2 - y_2| = 0$$

since $|x_1 - y_1| > 0$ and $|x_2 - y_2| > 0 \; \forall x \neq y$

$$\implies x_1 = y_1, x_2 = y_2 \implies x = y$$

2. condition: $d_1(x, y) = d_1(y, x)$.
We have that

$$d_1(x, y) = |x_1 - y_1| + |x_2 - y_2|$$

Now use the fact that $|a - b| = |b - a|$ and we get that

$$d_1(x, y) = |y_1 - x_1| + |y_2 - x_2| = d_1(y, x)$$

3. condition : $d_1(x, z) \leq d_1(x, y) + d_1(y, z)$ We start with the expression

$$d_1(x, z) = |x_1 - z_1| + |x_2 - z_2|,$$

and then subtract and add the first and second components of y inside the two absolute values respectively to get

$$d_1(x,z) = |(x_1 - y_1) + (y_1 - z_1)| + |(x_2 - y_2) + (y_2 - z_2)|.$$

Now using that $|a + b| \leq |a| + |b|$ to get the inequality

$$d(x,z) \leq |x_1 - y_1| + |x_2 - y_2| + |y_1 - z_1| + |y_2 - z_2| = d_1(x,y) + d(y,z).$$

□

Showing that $d_2$ is a metric.

1. condition: $d_2(x,y) = 0 \implies x = y$.

$$d_2(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

since $(x_1 - y_1)^2 > 0$ and $(x_2 - y_2)^2 > 0 \; \forall x \neq y$ and $(x_1 - y_1)^2 = (x_2 - y_2)^2 = 0$ if and only if $x = y$, then

$$d_2(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} = 0 \implies x_1 = y_1, x_2 = y_2 \implies x = y$$

2. condition: $d_2(x,y) = d_2(y,x)$.

$$d_2(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} = \sqrt{(-1)^2(y_1 - x_1)^2 + (-1)^2(y_2 - x_2)^2},$$

and since $(-1)^2 = 1$ we get

$$d_2(x,y) = \sqrt{(y_1 - x_1)^2 + (y_2 - x_2)^2} = d_2(y,x).$$

3. condition : $d_2(x,z) \leq d_2(x,y) + d_2(y,z)$
For simplicity and without loss of generality, let $y = (0,0)$. This simply corresponds to shifting the coordinate system. The challenge is then to show that

$$\sqrt{(x_1 - z_1)^2 + (x_2 - z_2)^2} \leq \sqrt{x_1^2 + x_2^2} + \sqrt{z_1^2 + z_2^2}.$$

Squaring both sides yields

$$x_1^2 + z_1^2 - 2x_1z_1 + x_2^2 + z_2^2 - 2x_2z_2 \leq x_1^2 + x_2^2 + z_1^2 + z_2^2 + 2\sqrt{x_1^2 + x_2^2}\sqrt{z_1^2 + z_2^2},$$

which simplifies to

$$-(x_1z_1 + x_2z_2) \leq \sqrt{(x_1^2 + x_2^2)(z_1^2 + z_2^2)}$$

If the left hand side of the inequality is less than or equal to zero, the the expression is obviously true, since the right hand side is always greater than or equal to zero. We then examine the case when the left hand side is greater than zero. This case allows us to square the inequality as follows.,

$$x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 \leq (x_1^2 + x_2^2)(z_1^2 + z_2^2).$$

This is then simplified to

$$0 \leq (x_1z_2 + x_2z_1)^2,$$

which also obviously is true, since the right hand side can not become negative.

The triangle inequality can also be proven by use of Minkowskis inequality and substituting $a_i = x_i - y_i$ and $b_i = y_i - z_i$ for $i = \{1,2\}$

$$\sqrt{\sum_{i=1}^{2}(a_i + b_i)^2} = \sqrt{\sum_{i=1}^{2}(x_i - z_i)^2} \leq \sqrt{\sum_{i=1}^{2}(a_i)^2} + \sqrt{\sum_{i=1}^{2}(b_i)^2} = \sqrt{\sum_{i=1}^{2}(x_i - y_i)^2} + \sqrt{\sum_{i=1}^{2}(y_i - z_i)^2}$$

We directly get the triangle inequality □

Showing that $d_\infty$ is a metric.

1. condition: $d_\infty(x,y) = 0 \implies x = y$. We have:

$$0 = d_\infty(x,y) = \max_{i\in 1,2}\{|x_i - y_i|\} \geq |x_i - y_i| \geq 0 \implies |x_i - y_i| = 0 \forall i\{1,2\} \implies x = y$$

2. condition: $d_\infty(x,y) = d_\infty(y,x)$.

$$|a-b| = |b-a| \implies d_\infty(x,y) = \max\{|x_1 - y_1|, |x_2 - y_2|\} = \max\{|y_1 - x_1|, |y_2 - x_2|\} = d_\infty(y,x)$$

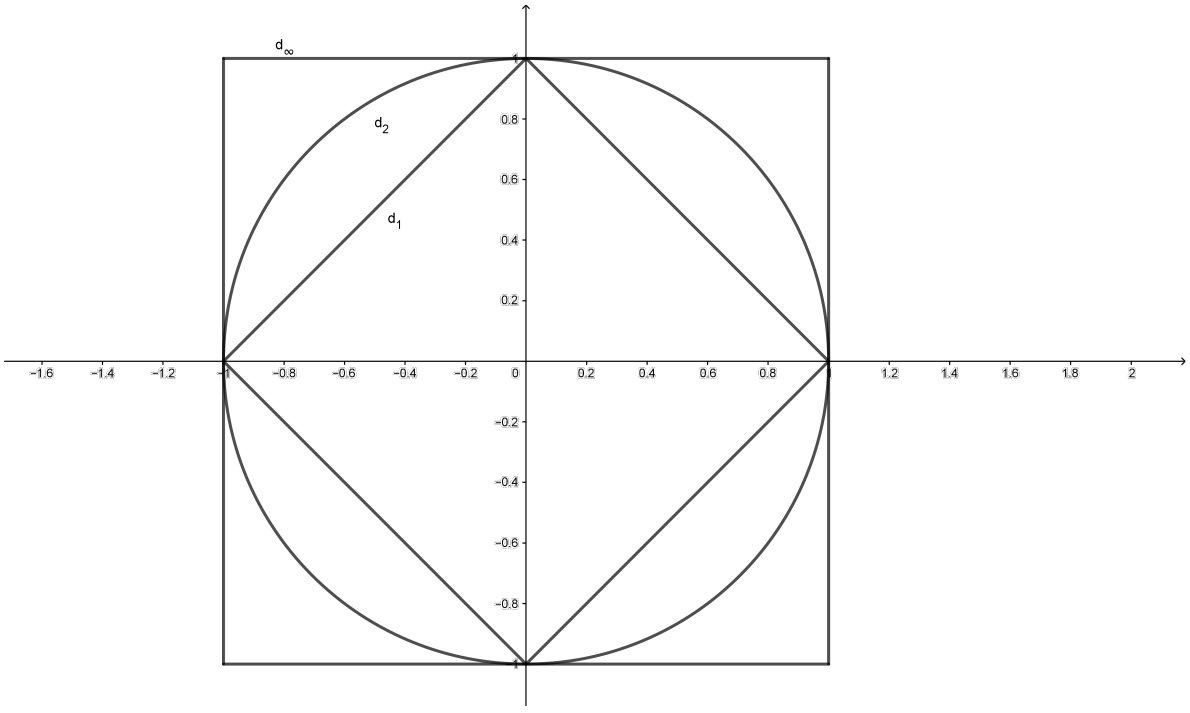
3. condition : $d_\infty(x,z) \leq d_\infty(x,y) + d_\infty(y,z)$

Using that $|a+b| \leq |a| + |b|$ we get,

$$\begin{aligned} d_\infty(x,z) = \max_i\{|x_i - z_i|\} = \max_i\{|x_i - y_i + y_i - z_i|\} &\leq \max_i\{|x_i - y_i| + |y_i - z_i|\} \\ &\leq \max_i\{|x_i - y_i|\} + \max_i\{|y_i - z_i|\} = d_\infty(x,y) + d_\infty(y,z) \end{aligned}$$

□

## 2.2 Task 2




## 2.3 Task 3

Want to show that all norms, $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$, are convex functions. $\forall x, y \in \mathbf{R}^n$ and $0 \leq \lambda \leq 1$ an arbitrary norm can be written as:

$$\|\lambda x + (1-\lambda)y\| \leq \|\lambda x\| + \|(1-\lambda)y\|$$

We get the inequality from the triangle inequality given in the definition of norms. Furthermore, from the positive homogeneity condition we get

$$\lambda\|x\| + \|(1-\lambda)y\| = |\lambda|\|x\| + |1-\lambda|\|y\| = \lambda\|x\| + (1-\lambda)\|y\|$$

which by the definition of convex functions, 1.1, defines a convex function.

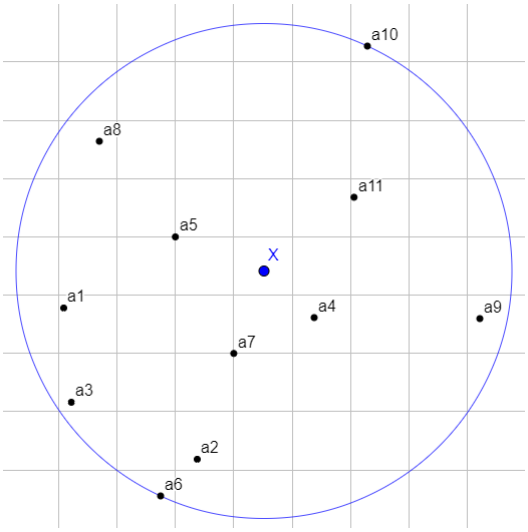



Figure 1: Visualisation of the minimization problem in Task 5

## 2.4 Task 4

Want to prove that

$$f(x) = \max_{a \in A} d(a, x) \tag{5}$$

$$g(x) = \sum_{a \in A} d(a, x) \tag{6}$$

are convex functions. We are going to prove this for the following metrics $d_1, d_2$ and $d_\infty$. Since $d_i(x, y) = \|x - y\|_i$ , $i \in \{1, 2, \infty\}$ and every norm is convex (from task 3) then, by equation (2) and (3), (5) and (6) must be convex as well.

## 2.5 Task 5

Solving the problem

$$\min_{x \in R^2} \max_{a \in A} d_2(a, x) \tag{7}$$

can be viewed as the geometrical problem of finding the smallest circle covering all points $a \in A$. The center point of which will be the unique minimizer of (7), and the function value of the objective function is the radius. See figure 1.

## 2.6 Task 6

Considering the problem

$$f(x) = \min_{x \in R^2} \sum_{a \in A} d_1(a, x) \tag{8}$$

we can separate the objective function in terms of its variables and solve for each individually.

$$\min_{x \in R^2} \sum_{a \in A} d_1(a, x) = \min_{x \in R^2} \sum_{a \in A} (|a_1 - x_1| + |a_2 - x_2|) = \min_{x \in R^2} \sum_{a \in A} |a_1 - x_1| + \min_{x \in R^2} \sum_{a \in A} |a_2 - x_2|$$

$$\frac{\partial f(x)}{\partial x_i} = \sum_{a \in A} \frac{x_i - a_i}{|x_i - a_i|}, \text{ where } i = \{1, 2\}$$

Similarly to solving the median problem for the euclidian norm, one has to implement some kind of numerical method to solve each of the one-dimensional minimization problems, such as a fixed point iteration.

## 2.7 Task 7

Want to prove that

$$h(x) = \sum_{a \in A} (d_2(a,x))^2 = \sum_{a \in A} ((a_1 - x_1)^2 + (a_2 - x_2)^2) \tag{9}$$

is convex. In the lectures we have proven that a twice differentiable function is convex if and only if its hessian is positive definite $\forall x \in \mathbf{R}^2$.

$$\frac{\partial h(x)}{\partial x_i} = -2 \cdot \sum_{a \in A} (a_i - x_i) \text{ where } i = \{1, 2\}$$

$$\nabla^2 h(x) = \begin{pmatrix} 2 \cdot n & 0 \\ 0 & 2 \cdot n \end{pmatrix} = 2n\mathbf{I}$$

Where n is the number of elements in A. Since the identity matrix is positive definite, it is also clear that the hessian is positive definite for all $x \in \mathbf{R}^2$. Thus, h(x) is convex and we have a unique solution. The solution is easily found by setting the partial derivatives equal to zero:

$$\nabla h(x) = 0 \implies x = \frac{1}{n} \sum_{j=1}^{n} a_j$$

## 2.8 Task 8

We have the minimization problem

$$\min_{x \in \mathbf{R}^2} f(x) \text{ where } f(x) = \sum_{a \in A} d_2(a,x) = ((a_1 - x_1)^2 + (a_2 - x_2)^2)^{\frac{1}{2}} \tag{10}$$

From 2.4 we have that the objective function is convex. The gradient is

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \end{pmatrix} \tag{11}$$

$$\frac{\partial f(x)}{\partial x_i} = \sum_{a \in A} \frac{x_i - a_i}{((a_1 - x_1)^2 + (a_2 - x_2)^2)^{\frac{1}{2}}}, \text{ where } i = \{1, 2\}$$

Since the gradient is undefined if x is an element of A we have to find the minimum of two separate minimization problems. The new minimization problem is:

$$\min_{x \in \mathbb{R}^2} f(x) = \min \begin{cases} \min_{x \notin A} f(x) \\ \min_{x \in A} f(x) \end{cases}$$

Looking at the minimization problem for $\min_{x \notin A f(x)}$. Let $x^*$ fulfill the first order necessary condition of $\nabla f(x) = 0$. Since it was proven in 2.4 that the objective function is convex, a stationary point is the global minimizer. Thus, $x^*$ is a minimizer of $f(x), x \in \mathbb{R}^2 \backslash A$. Since A is a finite set $\min_{x \in A} f(x)$ be solved by looking at each individual element of A.

## 2.9 Task 9

An interesting set of existing location, for which the solutions to (9) and (10) do not coincide, is a set where all the locations are collinear, they fall on a single line. Looking at the case of three points $a_1, a_2, a_3$ on the x-axis. With,

$$(a_2 - a_1) < (a_3 - a_2) < (a_3 - a_1) \tag{12}$$

The minimizer $x$ intuitively has to be $a_1 \leq x \leq a_3$ for both problems.

First we look at $\min_{x_1 \in R} \sum_i d_2(x_1, a_i)$:

$$\min_{x_1 \in R} \sum_i |x_1 - a_i| = \min_{x_1 \in R} (|x_1 - a_1| + |x_1 - a_2| + |x_1 - a_3|)$$

since $a_1 \leq x \leq a_3$, we get

$$\begin{aligned}\min_{x_1 \in R}(|x_1 - a_1| + |x_1 - a_2| + |x_1 - a_3|) &= \min_{x_1 \in R}(x_1 - a_1 + |x_1 - a_2| + a_3 - x_1) \\ &= \min_{x_1 \in R}(a_3 - a_1 + |x_1 - a_2|)\end{aligned}$$

and since $\min_{x_1} |x_1 - a_2|$ the unique solution to the minimization problem is $x_1 = a_2$.

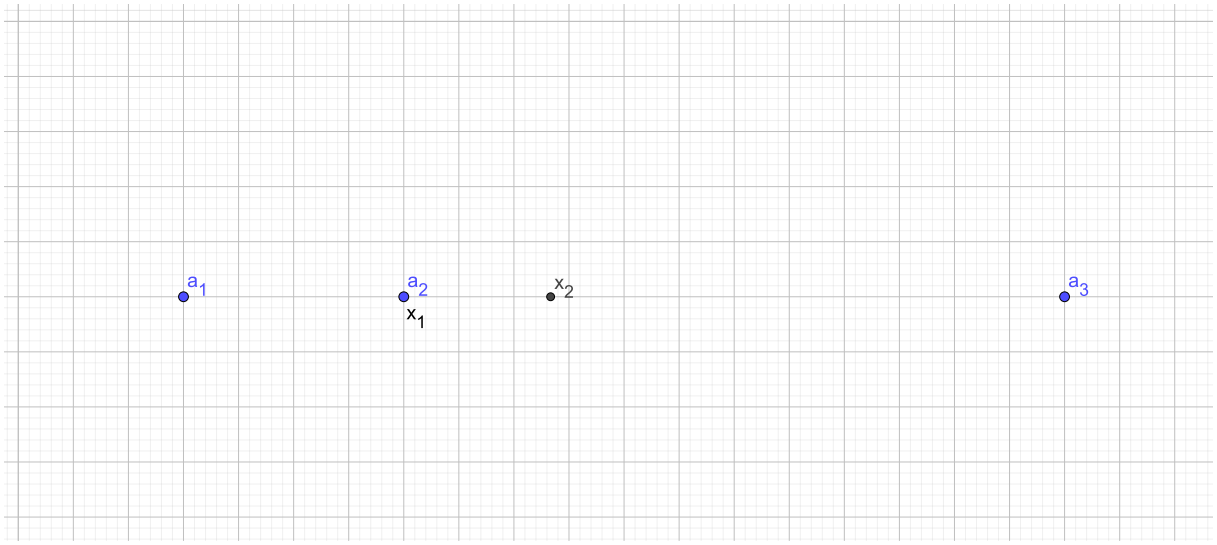
Then we look at $\min_{x_2 \in R} \sum_i (d_2(x_2, a_i))^2$:

$$\begin{aligned}\min_{x_2 \in R} \sum_i (x_2 - a_i)^2 &= \min_{x_2 \in R}((x_2 - a_1)^2 + (x_2 - a_2)^2 + (x_2 - a_3)^2) \\ &= \min_{x_2 \in R}(3x_2^2 - 2(a_1 + a_2 + a_3)x_2 + a_1^2 + a_2^2 + a_3^2)\end{aligned}$$

Since this is a quadratic polynomial with positive 2. degree term, the minimizer is found by solving the derivative = 0.

$$\begin{aligned}&6x_2 - 2(a_1 + a_2 + a_3) = 0 \\ &x_2 = \frac{1}{3}(a_1 + a_2 + a_3) \\ &x_2 = \frac{1}{3}(a_1 + a_2 + a_2 + (a_3 - a_2)) \\ &x_2 > \frac{1}{3}(a_1 + a_2 + a_2 + (a_2 - a_1)) \qquad * (12) \\ &x_2 > a_2\end{aligned}$$

Thus we have $x_1 \neq x_2$ since $x_1 = a_2$ and $x_2 > a_2$.




# 3 Project Tasks - Weiszfeld Algorithm

## 3.1 Task 1

Start by considering the first order necessary, and since $f_{d_2}$ is strictly convex, sufficient condition $\nabla f_{d_2}(x*) = \vec{0}$. We also assume that $x^*$ is a differentiable point.This gives rise to two equations.

$$\frac{\partial f_{d_2}}{\partial x_k} = \sum_{i=1}^{m} \frac{v^i(x_k - a_{i,k})}{d_2(x, a_i)} = 0, k = 1, 2.$$

Rearranging this expression we get an expression for $x$, dependent on $x$. If we let the right hand side express the respective component of $x$ in the next iteration, $x^{new}$, then we have the fixed point iteration scheme from (3.1) in the problem description:

$$x_k^{new} = \frac{\sum_{i=1}^{m} \frac{v^i a_{i,k}}{d_2(x,a_i)}}{\sum_{i=1}^{m} \frac{v^i}{d_2(x,a_i)}}, k = 1, 2.$$

A proof for the convergence of this iteration scheme is presented in [1].

### 3.2 Task 2

Proving Theorem 4 is simply a matter of interpreting the expression for $\frac{UB(x)}{LB(x)}$. $UB(x)$ is the upper bound on the value $f_{d_2}(x) - f_{d_2}(x^*)$ and $LB(x)$ is a lower bound on $f_{d_2}(x^*)$. Let relative accuracy be defined as

$$rel.ac. = \frac{f_{d_2}(x) - f_{d_2}(x^*)}{f_{d_2}(x^*)}.$$

We see that we have an upper bound on the numerator with $UB(x)$, and a lower bound on the denominator with $LB(x)$. Resulting in

$$f_{d_2}(x) - f_{d_2}(x^*) \leq UB(x),$$
$$f_{d_2}(x^*) \geq LB(x)$$
$$\implies rel.ac. = \frac{f_{d_2}(x) - f_{d_2}(x^*)}{f_{d_2}(x^*)} \leq \frac{UB(x)}{LB(x)}.$$

So if for a given $\epsilon > 0$ we have $\frac{UB(x)}{LB(x)} < \epsilon$, it follows that $rel.ac. < \epsilon$.

### 3.3 Task 3

Theorem 4 suggests the following termination criterion for the main iteration scheme in the Weiszfeld Algorithm: Given a relative accuracy threshold $\epsilon$ and a current point of the iteration $x$.

1. $if \frac{UB(x)}{LB(x)} \leq \epsilon \implies$ return $x$.
2. $else \implies$ Choose new $x$ according to (3.1) and return to 1.

When the algorithm terminates at $x$, the objective function is guaranteed to be within a factor of $(1+\epsilon)$ from $f(x^*)$.

$$f(x) \leq f_{d_2}(x^*) \cdot (1+\epsilon).$$

Furthermore, $||\nabla f_{d_2}(x)||$ can be somewhat simplified:

$$||\nabla f_{d_2}(x)|| = ||\nabla \sum_{i \in M} v^i d_2(a^i, x)|| = || \sum_{i \in M} v^i \nabla d_2(a^i, x)||.$$

For the Euclidean norm we have that

$$\nabla d_2(a^i, x) = \frac{x - a^i}{||x - a^i||},$$

which is a unit vector pointing from $a^i$ to $x$. Using this we get that

$$||\nabla f_{d_2}(x)|| = || \sum_{i \in M} v^i \frac{x - a^i}{||x - a^i||}||$$

This gives the following expression which easier to implement on a computer,

$$\frac{UB(x)}{LB(x)} = \frac{||\nabla f_{d_2}(x)|| \cdot \sigma(x)}{f_{d_2}(x) - ||\nabla f_{d_2}(x)|| \cdot \sigma(x)} = \frac{|| \sum_{i \in M} v^i \frac{x-a^i}{||x-a^i||}|| \cdot \sigma(x)}{f_{d_2}(x) - || \sum_{i \in M} v^i \frac{x-a^i}{||x-a^i||}|| \cdot \sigma(x)}.$$

We also observe that when calculating $\sigma(x)$, we only need to consider actual points in the set $M$. This is because $d_2(x, y) : y \in conv(a^1, ..., a^m)$ has to take its maximum value where $y$ is identical to some $a^i$.

This is a good stop criteria since you can be really confident in the accuracy of the minimizer you obtain.

The problem however is that it is expensive to calculate. To calculate the stop criteria we need to loop through all elements of $M$ which is a lot of calculations compared to other possible stop criteria. Mainly it is the fact that the gradient isn't already used in the iteration scheme that makes this an expensive termination criterion. We will discuss this further in the next section.

### 3.4 Task 4 and 5

We start by showing the pseudo-code for the two algorithms. The first eight lines are identical. They both check if any existing points are minimums by the given criterion. If one or more existing points satisfy the condition, the algorithms returns these points and never starts the main iterations. This check is motivated by the fact that the gradient is not defined in existing points. This fact might also cause problems in the the rest of the algorithm, if after an iteration we land in an existing point. We therefore add a few lines of code that check for this case and handles it, even though it might be a very rare case. When handling these situations we can safely assume that the anchor point isn't a minimum since our algorithm checks all of them first. Our pruposed solution is simply to increase the step size by some small amount $\theta$ to get out of the anchor point. In the case were the chosen starting point, in our case the median, is in a existing point, we choose a random new starting point near the original.

Both algorithms use the solution to the squared median problem as their starting point. Our main implementation of the Weiszfeld algorithm used the termination criterion formulated in 3.3. For the gradient descent with backtracking algorithm we implemented it with steepest descent as the search direction and the euclidean norm of the gradient as the termination criterion.

**Algorithm 1** Weiszfeld Algorithm

Given unique locations $a^i = (a_1^i, a_2^i)^T \in \mathbf{R}^2$, $v^i > 0$ for $i \in M = (1, 2, ..., m)$, some small $\theta$ and error tolerance $\hat{e}$.

**for all** $k \in M$ **do**
  $Test_k \leftarrow \sqrt{(\sum_{i \in M/k} v^i \frac{a_1^k - a_1^i}{d_2(a^k, a^i)})^2 + (\sum_{i \in M/k} v^i \frac{a_2^k - a_2^i}{d_2(a^k, a^i)})^2}$
  **if** $Test_k \leq v_k$ **then**
    *add* $a^k$ *to list of minimums*
  **end if**
**end for**
**if** *list of minimums is not empty* **then**
  **return** *list of minimums*
**end if**
$e^{j=0} \leftarrow \hat{e} + 1$
$x_1^{j=0} \leftarrow average_{i \in M}(a_1^i)$
$x_2^{j=0} \leftarrow average_{i \in M}(a_2^i)$
**while** $x^0 \in M$ **do**
  *choose some new starting location close to previous* $x^0$
**end while**
**while** $e > \hat{e}$ **do**
  $x_1^{j+1} \leftarrow \frac{\sum_{i=1}^m \frac{v^i a_1^i}{d_2(a^i, x)}}{\sum_{i=1}^m \frac{v^i}{d_2(a^i, x)}}$
  $x_2^{j+1} \leftarrow \frac{\sum_{i=1}^m \frac{v^i a_2^i}{d_2(a^i, x)}}{\sum_{i=1}^m \frac{v^i}{d_2(a^i, x)}}$
  $||\nabla f_{d_2}^j|| \leftarrow ||\sum_{i \in M} v^i \frac{x - a^i}{||x - a^i||}||$
  $\sigma^j \leftarrow \max_{y \in conv(a^1, ..., a^m)} (d_2(x^j, y)$
  $e \leftarrow \frac{||\nabla f_{d_2}^j|| \cdot \sigma^j}{f_{d_2}(x) - ||\nabla f_{d_2}^j|| \cdot \sigma^j}$
  **while** $x^j \in$ M **do**
    $x^j \leftarrow x^j + \theta \cdot (x^j - x^{(j-1)})$
  **end while**
  $j \leftarrow j + 1$
**end while**
**return** $(x_1^j, x_2^j)$

In the two figures, figure 2 and figure 3, we see the sequence of iterations for Weiszfeld and gradient descent respectively for the same problem instance. The problem consists of 20 points randomly generated integer points in a 100x100 square. The relative accuracy we demand from the Weiszfeld algorithm in this example is $e = 0.001$, and for the gradient descent we set $tol = 0.01$. With these criteria Weiszfeld completed 21 iterations before terminating. The gradient descent method only completed 11 iterations, but were closer to the result that the Weiszfeld algorithm gave after 60 iterations, than Weiszfeld itself

**Algorithm 2** Gradient Descent Algorithm

Given unique locations $a^i = (a_1^i, a_2^i)^T \in \mathbf{R}^2$, $v^i > 0$ for $i \in \mathbf{M} = (1, 2, ..., m)$, some small $\theta$ and termination tolerance $tol$. Backtracking parameters: $\hat{\alpha} > 0$, $\rho \in (0, 1)$ and $c \in (0, 1)$.

**for all** $k \in$ M **do**
  $Test_k \leftarrow \sqrt{(\sum_{i \in M/k} v^i \frac{a_1^k - a_1^i}{d_2(a^k, a^i)})^2 + (\sum_{i \in M/k} v^i \frac{a_2^k - a_2^i}{d_2(a^k, a^i)})^2}$
  **if** $Test_k \leq v_k$ **then**
    *add* $a^k$ *to list of minimums*
  **end if**
**end for**
**if** *list of minimums is not empty* **then**
  **return** *list of minimums*
**end if**
$x^{j=0} \leftarrow \frac{1}{m}(\sum_{i \in M} a_1^i, \sum_{i \in M} a_2^i)$
**while** $x^0 \in M$ **do**
  *choose some new starting location close to previous* $x^0$
**end while**
**while** $||\nabla f_{d_2}^j|| > tol$ **do**
  $\nabla f_{d_2}^j \leftarrow \sum_{i \in M} v^i \frac{x - a^i}{||x - a^i||}$
  $p^j \leftarrow -\nabla f_{d_2}^j$
  $\alpha^j \leftarrow \hat{\alpha}$
  **while** $f(x^k + \alpha \cdot \nabla f_{d_2}^k) > f_{d_2}(x^k) + c \cdot \alpha \cdot (\nabla f_{d_2}^j)^T p^k$ **do**
    $\alpha \leftarrow \alpha \cdot \rho$
  **end while**
  $x^{j+1} \leftarrow x^j + \alpha^j \cdot p^j$
  **while** $x^j \in$ M **do**
    $x^j \leftarrow x^j + \theta \cdot (x^j - x^{(j-1)})$
  **end while**
  $j \leftarrow j + 1$
**end while**
**return** $x^j$

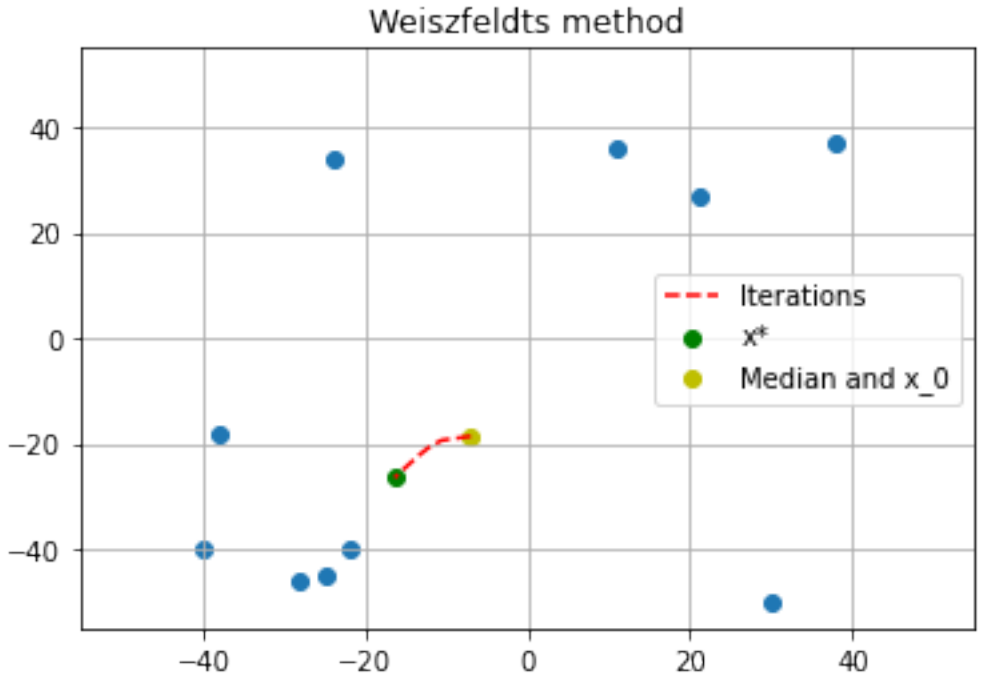



Figure 2: Weiszfeld solving a problem of 10 points and $\hat{e} = 0.001$ in 21 iterations.

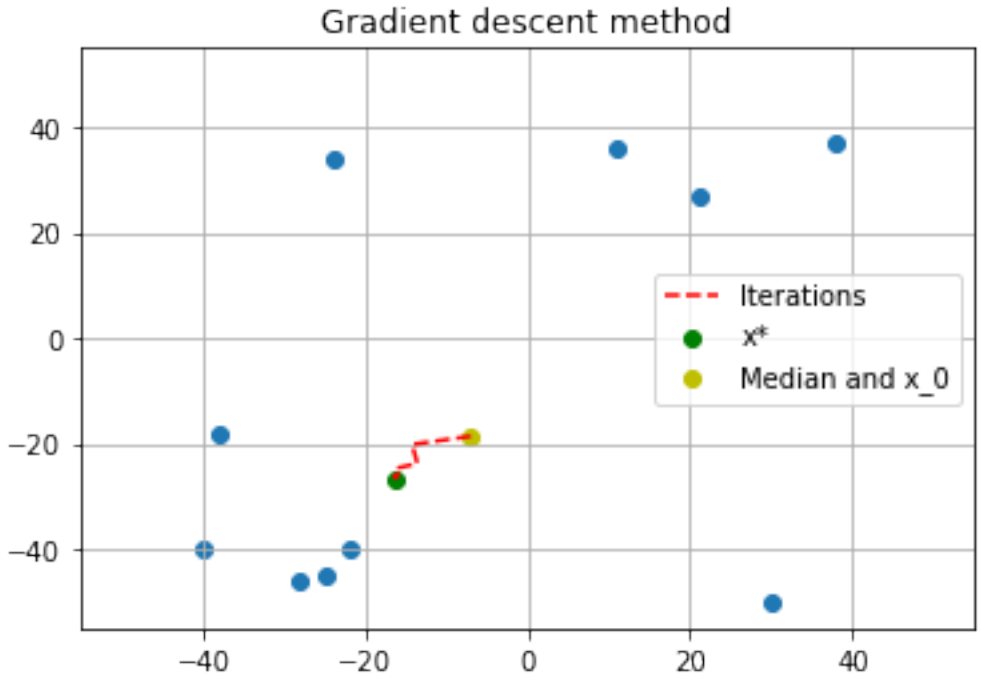



Figure 3: Steepest descent method solving a problem of 10 points and $tol = 0.01$ in 11 iterations.

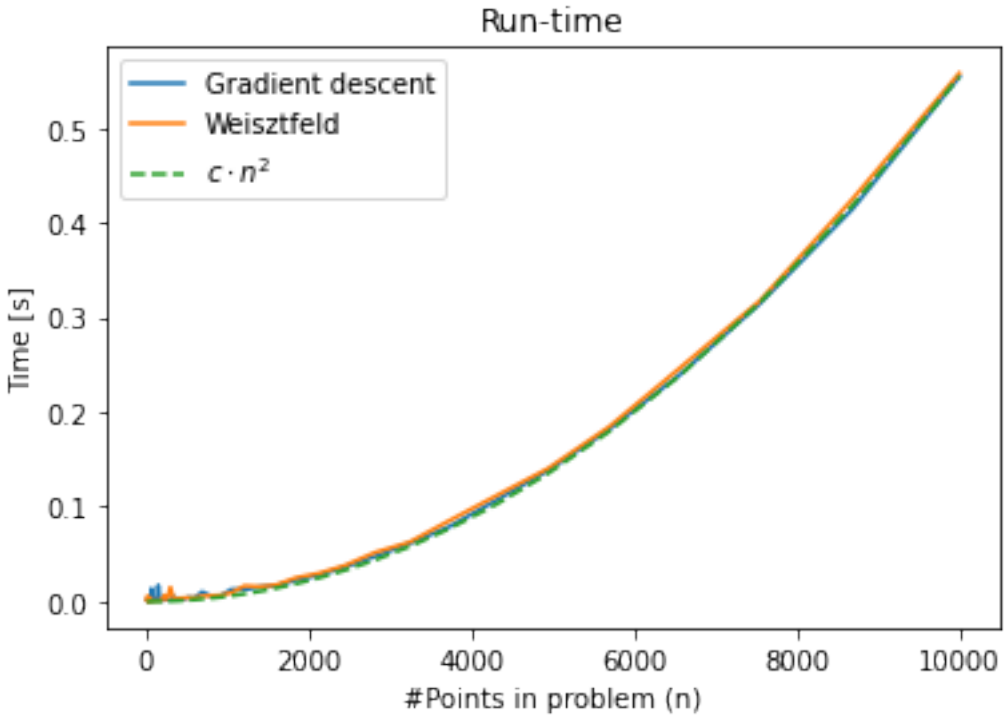



Figure 4: Running time analysis with number of points ranging from 100 to 10 000.

were after 21 iterations. This is not an optimal framework for analyzing the two algorithms. We therefore do a few changes before further discussing and testing their performance.

By testing the two algorithms on the same problems we are able somewhat to compare their performance. We also decided to implement the same termination criterion, $UB/LB < \epsilon$, in both Weiszfeld and gradient descent. Since the gradient descent method already utilizes the gradient, it should not add much extra computation time to use this termination criterion compared to using the norm of the gradient. Our numerical experiment worked as follows. For hundred different number of points evenly spaced on a logarithmic scale between hundred and ten thousand, we let both algorithms solve ten randomly generated problems limited to a ten thousand by ten thousand square. We then take the average run time over these ten problems. To improve the run time we used the numba package in python. The results are presented in figure 4. The result indicates, as expected, a time complexity of $n^2$. We note that the algorithms used fewer steps for bigger problem sizes. This is an effect of using relative accuracy and how we generated the problems. The points we generate are uniformly sampled from a pre-set interval. Due to symmetry, the expected value of $x^*$ is $(0, 0)$ with a decreasing variance for increasing problem sizes. This is also the case for our starting point, which is the solution to the median problem. With greater problem sizes the algorithm will start closer to the true minimum and the radius around the true minimum for which the relative accuracy is sufficient, increases.

The procedure of checking if each point anchor point is a minimizer, which is identical in both algorithms, takes $O(n^2)$ time. Analysing the run time of the rest of the algorithm becomes a bit more vague. For each iteration done by the Weiszfeld algorithm we calculate the gradient and next two coordinates of $x$, which bot take $O(n)$ time. In the gradient descent method, for each iteration there are two main procedures, calculating the gradient and doing the backtracking. Especially the backtracking procedure might be expensive as it involves multiple function evaluations which take $O(n)$ time each. We assume that the backtracking procedure takes less than $O(n)$ iterations. Since we have already established that the algorithm don't need more iterations for bigger problems, it is the procedure of testing all existing points that dominates the run time of $O(n^2)$. This is of course just analysing the time complexity and ignoring the procedures which take constant time and might separate the performance of the methods. In total we were unable to separate their performance in any significant way. There were instances where one out performed the other, but never reliably so.

We end by further discussing different potential termination criterion. In the gradient descent method, the termination criterion's discussed does not require much extra computation. Both checking the norm of the gradient and using $UB/LB > \epsilon$ involves using the gradient, which is already calculated. In Weiszfeld on the other hand, the termination criterion requires multiple extra summation loops, since it does not initially use the gradient in its iterations. We therefore propose an alternative termination criterion for the Weiszfeld algorithm. Use the change in $x$ over the last iteration. This does not require much extra computations.The downside to this is that we now loose the precise error bound at termination provided by the original termination criterion.

# References

[1] A. Beck and S. Sabach. "Weiszfeld's Method: Old and New Results". In: *Journal of Optimization Theory and Applications* 164 (2015), pp. 1–40. DOI: https://doi.org/10.1007/s10957-014-0586-7.