1 Nelder-Mead Simplex Method (Section 11.5)

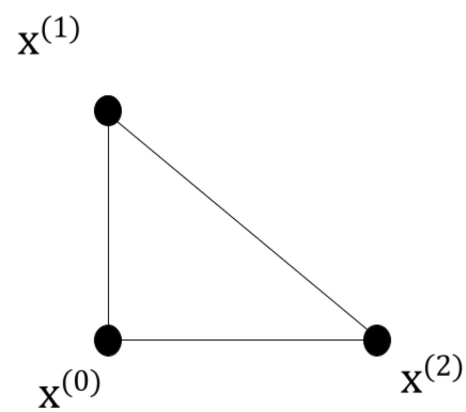
As usual, our goal is to minimize a function $f(x) : \mathbb{R}^n \to \mathbb{R}$. This method is a direct search method, but does not require a derivative of the function f and doesn't make use of any line searches. This method is versatile, but may be restricted to smaller problems[1].

Before beginning the calculations, we assign values to three parameters, $\alpha = 1$, $\beta = \frac{1}{2}$, and $\gamma = 1$. The values described here are the default values. In each step of this method, we'll have a set of $n+1$ points $x^{(i)} \in \mathbb{R}^n$,

$$\{x^{(0)}, x^{(1)}, \ldots, x^{(n)}\}.$$

Note that the set of vectors should be in general position in $\mathbb{R}^n$; that is, **the set of n points $x^{(i)} - x^{(0)}$ for $1 \le i \le n$ is linearly independent**.

(Example.) Suppose we're in $\mathbb{R}^2$. We have three points in our set, $x^{(0)} \in \mathbb{R}^2$, $x^{(1)} \in \mathbb{R}^2$, and $x^{(2)} \in \mathbb{R}^2$. This gives us



Here, we can consider $x^{(1)}$ and $x^{(2)}$ to be linearly independent, and $x^{(0)}$ is at the origin. By connecting these points with a border, we create a convex hull with the set of points. This convex hull represents a simplex.

We want to reorder/relabel the points so that

$$f(x^{(0)}) \ge f(x^{(1)}) \ge \ldots \ge f(x^{(n)}).$$

1.1 Operations on the Search Space

Based on the ordering, the search space will be modified in order to find the desirable points. As one might expect, $f(x^{(0)})$ is the worst and $f(x^{(n)})$ is the best. Some operations to explore the search space include

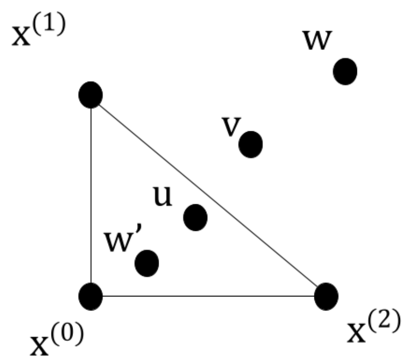
- **Centroid:** $u = \frac{1}{n}\sum_{i=1}^{n} x^{(i)}$. This is the centroid of the face of the current simplex opposite the worst vertex, $x^{(0)}$.
- **Reflection:** $v = (1+\alpha)u - \alpha x^{(0)} - \alpha x^{(0)}$
- **Expansion:** $w = (1+\gamma)v - \gamma u$. This checks if v was improving the best function value, whether w would improve it even more.
- **Reduction:** $w' = \beta x^{(0)} + (1-\beta)u$.

[1] In other words, for a small n.

(Example.) If we have $x^{(0)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $x^{(1)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, and $x^{(2)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, then we know $n = 2$ and

$$u = \frac{1}{n} \sum_{i=1}^{n} x^{(i)} = \frac{1}{2} \sum_{i=1}^{2} x^{(i)} = \frac{1}{2} \left(\begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) = \frac{1}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Roughly speaking, these operations look like



In particular, part of the check we need to do is whether $f(v) < f(x^{(n)})$. If this is the case, we can compute w (the expansion) and see whether this expansion is further beneficial. In particular, we want to see if $f(w) < f(x^{(n)})$; if this is the case, then we can set $x^{(0)} \leftarrow w$ and $f(x^{(0)}) \leftarrow f(w)$. Otherwise, we can set $x^{(0)} \leftarrow v$ and $f(x^{(0)}) \leftarrow f(v)$.

1.2 Desirable Stopping Condition

A desirable stopping condition is the **relative flatness**; that is,

$$\frac{f(x^{(0)}) - f(x^{(n)})}{\max\{|f(x^{(0)})| + |f(x^{(n)})|, 1\}}.$$

Note that we have the maximum condition in the denominator to avoid dividing by 0. So, we would stop if the "relative flatness" is sufficiently small enough.

1.3 Algorithm

The implementation we'll use does not reorder all the functions, as this is an expensive process (sorting in general is expensive). Instead, we only swap the relative indices. With this said, our function takes the following inputs:

- f, the function to evaluate.
- $\{x^{(0)}, x^{(1)}, \ldots, x^{(n)}\}$, the set of candidate points.
- ϵ, the tolerance.
- M, the maximum number of iterations.
- α, β, γ, the three parameters mentioned earlier.

Algorithm 1 Nelder-Mead Simplex Method

```
function NelderMeadSimplex(f, {x^{(i)}}, \epsilon, M, \alpha, \beta, \gamma)
    for i \leftarrow 1 to n+1 do        ▷ Corresponds to constructing simplex by defining values at each vertex.
        F(i) = f(x^{(i-1)})             ▷ F is the vector of all function values.
        X(:, i) = x^{(i-1)}             ▷ Storing the points into the matrix.
    end for
    (f_n, i_n) = \min(F)
    (f_0, f_1, i_0, i_1) = \max_2(F)
    o \leftarrow [1]_{n\times 1}         ▷ n \times 1 vector of 1's.
    k \leftarrow 0
    while (f_0 - f_n)/(\max\{|f_0| + |f_1|, 1\}) \geq \epsilon and k < M do
        u \leftarrow \frac{1}{n}\sum_{\ell=1}^{n} x^{(\ell)}
        v \leftarrow (1+\alpha)u - \alpha X(:, i_0)
        f_v = f(v)
        if f_v < f_n then
            w \leftarrow (1+\gamma)v - \gamma u
            f_w \leftarrow f(w)
            if f_w < f_n then
                X(:, i_0) \leftarrow w
                f_0 \leftarrow f_w
            else
                X(:, i_0) \leftarrow v
                f_0 \leftarrow f_v
            end if
        else
            if f_v \leq f_1 then
                X(:, i_0) \leftarrow v
                f_0 \leftarrow f_v
            else
                b \leftarrow f_0
                if f_v \leq f_0 then
                    X(:, i_0) \leftarrow v
                    f_0 \leftarrow f_v
                end if
                w \leftarrow \beta X(:, i_0) + (1-\beta)u
                f_w \leftarrow f(w)
                if f_w \leq b then
                    X(:, i_0) \leftarrow w
                    f_0 \leftarrow f_w
                else
                    for i \leftarrow 1 to n+1 do
                        if i \neq i_n then
                            X(:, i) \leftarrow \frac{1}{2}(X(:, i) + X(i, i_n))
                            F(i) \leftarrow f(X(:, i))
                            break
                        end if
                    end for
                end if
            end if
        end if
        (f_n, i_n) \leftarrow \min(F)
        (f_0, f_1, i_0, f_1) \leftarrow \max_2(F)
        k \leftarrow k + 1
    end while
end function
```

A few notes:

- In line 6, we're computing the index of the minimum point. So, f_n is the smallest function value and i_n is minimum index of said function value.
- In line 7, we're finding the two largest function values f_0, f_1 and the corresponding indices i_0, i_1. Here, $\max_k$ means to find the k highest elements.