Mathematics 2, Homework 4 **April 1st 2022**

This is not a joke.

This is a proper homework assignment.

Nelder-Mead method

1. Implement the two and three dimensional versions of Nelder-Mead method. Use a programming language of your choice.

One step beyond: Why fix the dimension? Surely you can pass the dimension as a parameter or better still compute it on the fly.

2. Qualitatively compare Nelder-Mead method with the methods you have implemented in HW2/3,4 (GD, Polyak GD, Nesterov GD, AdaGrad GD, Newton, BFGS) on

$$f(x, y, z) = (x - z)^2 + (2y + z)^2 + (4x - 2y + z)^2 + x + y,$$

$$f(x, y, z) = (x - 1)^2 + (y - 1)^2 + 100(y - x^2)^2 + 100(z - y^2)^2,$$

and

$$f(x, y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2.$$

These functions are taken from HW2/5a,b,c. For the Nelder-Mead method choose starting samples of different diameters, one of the starting sample points should match the one taken from HW 2.

Black box optimization

Let *xxxxxxxx* be your student Id. The three functions

$$f_{xxxxxxxx,i} : \mathbb{R}^3 \to \mathbb{R},$$

$i \in \{1, 2, 3\}$ are given in a black-box setting.

The archive `hw4_executables.zip` (contained in the HW4 dropbox) contains three (equivalent) command line executables `hw4_nix`, `hw4_mac`, and `hw4_win.exe`, running on three different OS families. Each executable expects five parameters on the standard input and, after a time consuming computation, outputs a real number to the standard output. Below is a sample call.

```
$ ./hw4_mac xxxxxxxx i 3.17 0.71 -1.55
```

Your student Id is the first parameter *xxxxxxxx*, the second parameter is an integer $i \in \{1, 2, 3\}$, and the remaining three parameters are real numbers.

Try to test the appropriate executable as fast as possible. Please report any problems should the above program not work as intended.

3. Find minima of functions

$$f_{xxxxxxxx,1}, \qquad f_{xxxxxxxx,2}, \qquad f_{xxxxxxxx,3}.$$

   Use sufficiently high precision.

   These problems are in theory unconstrained, you are guaranteed that none of the calls results in an overflow if real parameters lie in $[-10, 10]$.

Another step beyond: How would one use a gradient-descent based method in such a case. Which one is best suitable. Can you beat Nelder-Mead?

Local search study

Let $G = (V, E)$ be a graph. A *matching* in G is a set of edges $M \subseteq E(G)$, so that no vertex $v \in V(G)$ is incident with more than one edge from M. Review the basic matching related problems.

3. What is a maximal matching? What is a perfect matching? Can you compute a maximal matching greedily? Does every graph have a perfect matching?

Let us take matchings one step further. Given a graph G, let $w : E(G) \to \mathbb{R}$ be edge weights[1]. If M is a matching in G, then its *weight*, $w(G)$, is defined as the sum of weight of its edges

$$w(M) = \sum_{e \in M} w(e).$$

The MaximalWeightMatching problem can be defined as

> MaximalWeightMatching
>
> **input:** Graph G, edge weights w.
>
> **output:** Matching M in G for which $w(M)$ is maximal.

In the next step we shall relax the above problem. If v is a vertex of G, let $E(v)$ denote the set of edges incident with v. $M \subseteq E(G)$ is a matching, if for every vertex v the intersection of M and $E(v)$ contains at most one edge.

Instead of putting an edge e in the matching M or not, let us consider vectors $x \in [0, 1]^E$. Such x is called a *fractional matching* if for every vertex v we have $\sum_{e \in E(v)} x_e \leq 1$.

4. Can you express every fractional matching as a convex combination of matchings.

Now the linear relaxation of the MaximalWeightMatching can be defined as

> relaxMaximalWeightMatching
>
> **input:** Graph G, edge weights w.
>
> **output:** Fractional matching x for which $\sum_{e \in E(G)} x(e)w(e)$ is maximal.

[1] Most often we require that weights are nonnegative, but we can do with a more general definition.

Note that relaxMaximalWeightMatching is a linear program.

Let G_{20} be a 20×20 grid graph — the vertex set consists of integral points in the plane with coordinates between 1 and 20, vertices at distance 1 being adjacent. Choose and fix a choice of random edge weights

$$w_{20} : E(G_{20}) \to [1, 2].$$

5. Find a solution of relaxMaximalWeightMatching with input G_{20}, w_{20} using a commercial LP solver[2].

6. What is the optimal fractional matching x^*? Inspect and comment.

Now let us play with local search. Given a matchings M_1 and M_2 in G we say that they are *k-adjacent* if M_2 can be obtained from M_1 by first removing $\leq k$ edges and replacing them with an alternative[3] collection of $\leq k$ edges. So, M_1 and M_2 are 1-adjacent if either

- $M_1 = M_2$ or
- M_2 can be obtained from M_1 by removing a single edge or
- M_2 can be obtained from M_1 by adding a single edge or
- M_2 can be obtained from M_1 by first removing and then adding an alternative edge.

7. Perform local optimization trying to solve MaximalWeightMatching with G_{20} and w_{20} as your input. Start with empty matching as your initial solution. Try at least 1- ,2- and 3-adjacency. Try removing random edges or edges with relatively low weight. Add either random edges or edges with relatively high weight. Accept a step if it does not decrease the weight of your matching.

8. Can you think of some sensible jump moves?

9. How close to x^* can you get?

Upload your solution in a single .zip archive which contains the source-code and a .pdf.

[2] You got familiar with LP solvers in Homework 3, right.

[3] Could be the same.