Computational assignment 8

due on November 17, 2025

Fluence map optimization with quadratic penalty functions

The next goal is to implement an IMRT fluence map optimization algorithm using gradient descent for the case that only quadratic penalty functions are used as objective functions and no dose constraints are used. Hence, we consider the objective function

$$f(d) = \sum_i w_i^o \left(d_i - D_i^{max}\right)_+^2 + \sum_i w_i^u \left(D_i^{min} - d_i\right)_+^2 \tag{1}$$

where

D_i^{max} is the maximum dose to voxel i,
D_i^{min} is the minimum dose to voxel i,
w_i^o is the overdose penalty factor for voxel i,
w_i^u is the underdose penalty factor for voxel i

The subscript + indicates that $(d)_+ = d$ if $d > 0$ and $(d)_+ = 0$ if $d \leq 0$. Thus, the first term quadratically penalizes doses d_i that exceed a maximum dose D_i^{max} but the penalty function is set to zero if the dose is lower than D_i^{max}. Analogously, the second term penalized doses below a minimum dose D_i^{min}.

For OARs, D_i^{min} or w_i^u would be zero. In general, penalty factors and maximum/minimum doses could vary on a voxel-by-voxel basis. However, here we can set these parameters such that they vary between different organs, but are constant between voxels belonging to the same organ. Typically, the penalty factors for each organ are scaled with the number of voxels in that organ.

Create a python dictionary `TPopt` that holds all parameters to specify a treatment plan optimization problem. Define maximum and minimum doses for all organs as well as penalty factors. For example:

```
TPopt['vois'][1] = {
    'name': 'normal tissue',
    'nVoxels': np.sum(voi == 1),
    'maxdose': 50,
    'overdosepenalty': 1,
    'mindose': 0,
    'underdosepenalty': 0,
}
```

Define vectors that are the same size as a dose vector for D_i^{max}, D_i^{min}, w_i^o, and w_i^u. For example:

```
TPopt['maxdose']
TPopt['mindose']
TPopt['overdosepenalty']
TPopt['underdosepenalty']
```

Write a function

`calculate_quadratic_objective(bixelweights)`

that returns the value of the objective function, given a vector of beamlet intensities as input.

Write a function

`calculate_quadratic_objective_gradient(bixelweights)`

that returns the gradient of the objective function, given a vector of beamlet intensities as input.

Gradient descent optimization algorithm

Minimization of the quadratic penalty function can be performed via a gradient projection algorithm which iteratively improves the beamlet intensities. In the initialization step, you have to choose a step size and initialize the beamlet intensities to zero:

0. Initialize the beamlet intensities $F_j = 0 \; \forall j$
   Choose a step size α.

The algorithm consists of iterating the following steps until convergence:

1. Given the beamlet intensities F^k of the current interation k, calculate the gradient
$$\left.\frac{\partial f}{\partial F_j}\right|_{F^k} \tag{2}$$
   of the objective function with respect to the beam intensities.
2. Update the beamlet intensities according to
$$F_j^{k+1} = F_j^k - \alpha \left.\frac{\partial f}{\partial F_j}\right|_{F^k} \tag{3}$$
3. Project negative beamlet intensities onto the non-negativity constraint, i.e. set $F_j = max(0, F_j)$.

Implementation:

Write a function

```
optimize_gradient_descent(nIterations, stepsize)
```

which implements this algorithm. The function takes a number of iterations and a step size as input and returns the vector of optimized beamlet intensities.

Treatment planning

Use the algorithm to create treatment plans. A typical tumor prescription dose for the treatment of spinal metastases would be 35 Gy delivered in 5 fractions. The dose distribution within the tumor should be relatively homogeneous. The dose to the spinal cord should be limited to 20 Gy. In addition, the dose distribution should be conformal, i.e. the volume outside of the tumor that receives 35 Gy should be small and the dose should fall off as steeply as possible with distance from the tumor. The dose received by the lungs and the esophagus should be as low as possible. Experiment with different values for D_i^{max}, D_i^{min}, w_i^o, and w_i^u to see how the dose distribution changes.

Hints

You may have to experiment with the step size to find a suitable value by trial and error. It is good to start with a small value (for me 0.1 worked) and check that the objective function value decreases in every iteration. You can then try to increase the step size in order to speed up convergence. For a step size that is too large, the algorithm will fail (i.e. the objective function value will not decrease in every iteration). The number of iteration should be large enough in order to converge to a good plan (probably in the order of several hundred to a thousand, depending on the step size). The algorithm can be stopped if the improvement in the objective function in one iteration becomes very small compared to the improvement during the first iterations.