Visual Scene Analysis
Institute for Computer Science
University of Siegen

Weekly Exercises 2

Room: H-C 7326
Wednesday, 02.11.2016, 14:15-15:45
Submission deadline: Monday, 31.10.2016, 16:00, CG box in front of room H-A7115
Programming: email to jonas.geiping@uni-siegen.de

Theory

Exercise 1 (4 points)**.** In this exercise we'd like to determine the shortest path $\phi : [0,1] \to \mathbb{R}^2$ from a point $a \in \mathbb{R}^2$ to a point $b \in \mathbb{R}^2$, i.e. , $\phi(0) = a$, $\phi(1) = b$. Without restriction of generality you may assume that $a_1 \leq b_1$, and you may assume without a proof that it never makes sense to "go backwards" on the x-axis. Mathematically, the latter means that we may reduce our problem to finding the *graph* of a 1D function. In other words, we may parametrize the desired shortest path ϕ as

$$\phi(x) = (xb_1 + (1-x)a_1, f(x)) \tag{1}$$

and look for the unknown 1D function $f : \mathbb{R} \to \mathbb{R}$.

- The length of a path $\phi : [0,1] \to \mathbb{R}^2$ is given by

  $$l(\phi) = \int_0^1 |\phi'(x)| \, dx = \int_0^1 \sqrt{\phi_1'(x)^2 + \phi_2'(x)^2} \, dx.$$

  Show that

  $$l(f) = \int_0^1 \sqrt{c + f'(x)^2} \, dx$$

  by using (1) and also that $c = (b_1 - a_1)^2$.

- Consider the shortest path problem in terms of your new variable f, i.e.

  $$\hat{f} = \text{argmin}_f \; l(f).$$

  Determine an optimality condition using the Euler-Lagrange equations!

- Conclude that the derivative of f must be constant.

You have successfully proven that the shortest path between two points is a line!

Exercise 2 (2 points)**.** Think of the discretization of the problem in exercise 1. Assume you discretize f at $n+2$ equidistant points $0 = x_0, x_1, ..., x_n, x_{n+1} = 1$. You know that $f_0 = f(x_0) = a_2$ and $f_{n+1} = f(x_{n+1}) = b_2$, so you only have n variables. Which discrete energy do you want to minimize to implement exercise 1? What is the gradient of your energy in the discrete case?

Programming

Exercise 3 (4 points)**.** Implement the gradient descent algorithm with backtracking in Matlab and test it on the shortest path problem!

Bonus Exercise 1 (4 extra points)**.** Write a MATLAB class "energy" that consists of two function handles - evaluating the energy at a specific point and evaluating the gradient of the energy at a specific point. Overload the "+" operator in order to be able to add two energies. Then write your implementation of the gradient descent algorithm with backtracking in such a way that it accepts an "energy" as an input.

Congratulation! You now have a very general tool for optimizing smooth convex energies! On the upcoming homework sheets you only have to specify your energy and run your existing code!