Exercises for
Introduction to Pattern Recognition (IntroPR)
Simone Gaffling and Oliver Taubmann
Work sheet 11, 19.01.2015

Nonlinear Optimization

Exercise 28 In the lecture, you have heard about descent methods for optimization. Some can be derived by looking at Taylor series approximations.

(a) **Gradient Descent.** Using the first order Taylor expansion of a scalar-valued, multivariate function $f(\boldsymbol{x})$ around the position $\boldsymbol{x}_i$, find the search direction $\boldsymbol{d}$, $\|\boldsymbol{d}\|_2 = 1$, that minimizes $f(\boldsymbol{x}_i + \boldsymbol{d})$. Which assumption does this approximation make about the function? What does it mean for optimization?

(b) **Newton Method.** Find a better search direction by taking the curvature of the function into account as well, i. e. include the second order term in the Taylor expansion. Can you think of reasons that might prevent us from using this (theoretically more accurate) approximation?

Exercise 29 **Programming Task:** Using nonlinear iterative optimization, find the position of the minimum $f(\boldsymbol{x}^*) = 0$, $\boldsymbol{x}^* = (1, 1)^\top$, of the Rosenbrock function, a simple bivariate function used for testing numerical optimizers. You can download an implementation of the function from the website (`rosenbrock.zip`). While MATLAB offers functions for this task, such as `fminunc`, we want to write an optimizer ourselves to discover how it works "under the hood."

- For visualizing the function, the MATLAB commands `surf` and `contour` might prove helpful. You may want to plot the path your optimization took, i. e. the sequence of estimates, on top.
- Initialize the process with different starting positions, e. g. $\boldsymbol{x}_0 = (-1.5; -1.5)^\top$, $\boldsymbol{x}_0 = (0.65; 0.75)^\top$, $\boldsymbol{x}_0 = (0.75; 1.5)^\top$ and $\boldsymbol{x}_0 = (-1.1; 2.0)^\top$. What do you notice? What implications does this have for nonlinear optimization in general?
- There are many ways to determine a search directions. Your program should allow choosing between different such strategies. For instance, it could offer the gradient descent and Newton methods as derived above. As for even simpler approaches, try going along the axes only (coordinate descent), or choose directions randomly. Make sure that your direction $\boldsymbol{d}$ satisfies $\boldsymbol{d}^\top \nabla f(\boldsymbol{x}_i) < 0$; if it does not, flip it. For gradient descent, also try approximating the gradient by central differences. Experimentally compare the speed of convergence of all methods and the paths they describe in the parameter space. What do you observe?
- Determining how far to walk along the search direction is an 1-D optimization problem in itself. Typically, this so called line search is solved heuristically. An approximate backtracking line search could start at $\boldsymbol{x}_i + t \cdot \boldsymbol{d}$, $t = 2$, and iteratively reduce t by 20 % as long as $f(\boldsymbol{x}_i + t \cdot \boldsymbol{d})$ is above the line $f(\boldsymbol{x}_i) + \frac{1}{2} \cdot t \cdot \boldsymbol{d}^\top \nabla f(\boldsymbol{x}_i)$.
- Stop iterating once the gradient magnitude $\|\nabla f(\boldsymbol{x}_i)\|_2$ falls below $\varepsilon = 10^{-3}$.