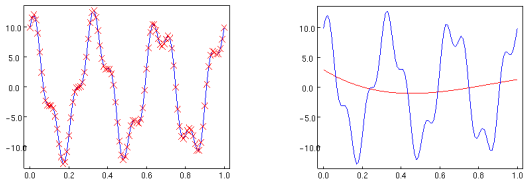
Lab Course (Basics & Lab)

Universität Bonn
Institut für Informatik II
July 12, 2016

Summer term 2016
Prof. Dr. Reinhard Klein
Stefan Krumpen, Alexander Dieckmann

Sheet 12 - Function Fitting

Practical assignments are corrected in the exercises on *Mo 25/7 14:30*
Single submissions!



Assignment 1 (Function fitting (convex), *4 points*)

Given a set of points $(x_i, y_i) \in \mathbb{R}^2$ we want to select a function $f_a : \mathbb{R} \to \mathbb{R}$ approximating them well. The function f_a is chosen from a family of functions $\{f_a\}$ where different parameters $a \in \mathbb{R}^n$ lead to different functions f_a. We want to choose a such that f_a approximates our points well. This is measured by the following energy functional:

$$e(a) := \sum_i (f_a(x_i) - y_i)^2.$$

Its value is the sum of the squared errors at the positions x_i.

To find the best approximation, we need to find the parameters $\hat{a} = \arg\min_a e(a)$ that minimize the energy. Most methods will rely on a gradient of our energy functional

$$\nabla e(a) = \sum_i 2 \cdot (f_a(x_i) - y_i) \cdot \begin{pmatrix} \frac{\partial f_a(x_i)}{\partial a_0} \\ \vdots \\ \frac{\partial f_a(x_i)}{\partial a_n} \end{pmatrix}.$$

The function that we want to approximate is $g_{3,10}$ with

$$g_{c_1,c_2} : x \mapsto c_1 \sin(2\pi c_2 x) + c_2 \cos(2\pi c_1 x)$$

a) Write a function `void sampleG()` which samples point from $g_{3,10}$ and adds them to global variables `vector<double> samplesX` and `sampleY`:

$$x_i = i/100.0 \qquad\qquad y_i = g_{3,10}(x_i)$$

b) Compute and plot the sample points $(x_0, y_0), \ldots, (x_{100}, y_{100})$ using the `SimpleGraph` interface from the previous assignment sheet.

c) Next we want to approximate the original function with a 3rd order polynomial $f_a(x) := a_3 x^3 + \ldots + a_1 x^1 + a_0$ specified by its coefficients $a := (a_3, \ldots, a_0)^t$. To calculate a good choice for those coefficients minimize the respective energy $e(a)$. Implement a function `polynomial_fit_energy()` to map a vector $a \in \mathbb{R}^4$ to its energy $e(a)$ and its gradient $\nabla e(a)$.

d) Output your gradient at the points: $a = (1,0,0,0)^t$, $a = (0,1,0,0)^t$, $a = (0,0,1,0)^t$, $a = (0,0,0,1)^t$ and $a = (4,3,2,1)^t$

Hint: The partial derivatives of the polynomials are $\frac{\partial f_a(x)}{\partial a_i} = x^i$. Gradients should be:

$$\begin{pmatrix}11.9\\22.3\\40.2\\31.0\end{pmatrix}, \begin{pmatrix}16.6\\28.9\\50.2\\47.7\end{pmatrix}, \begin{pmatrix}23.3\\38.9\\66.9\\81.0\end{pmatrix}, \begin{pmatrix}33.3\\55.6\\100.2\\182\end{pmatrix}, \begin{pmatrix}336.7\\418.0\\552.6\\791.0\end{pmatrix}$$

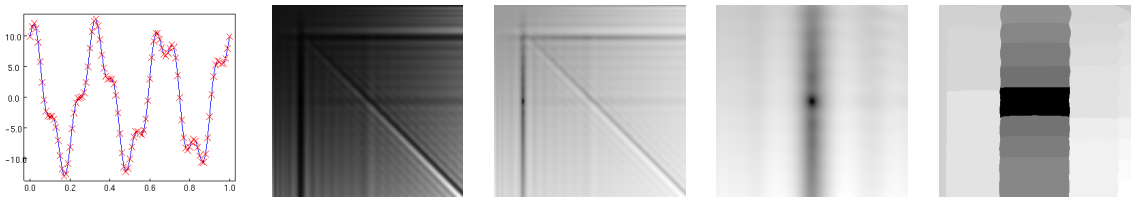
e) Pass the function `polynomial_fit_energy()` to `gradient_descent_nd()`. Start with the initial parameters $a_0 = \cdots = a_3 = 0$. Minimize $e(a)$. What are the final values of $a_0, \ldots, a_3$ and $e(a)$? Plot the functions $g_{3,10}$ and f_a in a common graph.

Hint: A correct implementation of the gradient descent should converge within ca. 50,000 iterations.

f) Now start with 10 different random vectors $a \in [0, 10000.0]^4$ as initial parameters a_i. What are the final values of a_i and $e(a)$? Given enough iterations, does it always converge to the best approximation? Does the problem have a 'global optimum'?

Hint: You can generate random vectors using `createRandom()` found in `tools_12.h`. Use ten different subsequent seeds starting at 123456.

Assignment 2 (Function fitting (non-convex), *4 points*)



Similar as in the last assignment we want to approximate $g_{3,10}$ with

$$g_{c_1,c_2} : \mathbb{R} \to \mathbb{R} \qquad g_{c_1,c_2} : x \mapsto c_1 \sin(2\pi c_2 x) + c_2 \cos(2\pi c_1 x)$$

by minimizing the energy

$$e(a) := \sum_i (f_a(x_i) - y_i)^2.$$

We already know that the function we are looking for is from $\{g_{c_1,c_2} \mid c_1, c_2 \in \mathbb{R}\}$. So instead of fitting a polynomial we will try to find the right parameters c_1, c_2.

a) Reuse your function `void sampleG()` which samples 100 point from $g_{3,10}$ and adds them to global variables `vector<double> samplesX` and `samplesY`:

$$x_i = i/100.0 \qquad y_i = g_{3,10}(x_i) \qquad i = 1, \ldots, 100$$

b) Implement a function `g_fit_energy()` to compute the energy E_{c_1,c_2} that describes the error of approximating the sampled points with g_{c_1,c_2}:

$$E : \mathbb{R}^2 \to \mathbb{R} \quad (c_1, c_2) \to \sum_{i=1}^{100} (g_{c_1,c_2}(x_i) - y_i)^2.$$

The gradient ∇E is $\begin{pmatrix} \frac{\partial E(c_1,c_2)}{\partial c_1} \\ \frac{\partial E(c_1,c_2)}{\partial c_2} \end{pmatrix}$ with

$$\frac{\partial E(c_1,c_2)}{\partial c_1} = \sum_{i=1}^{100} 2(g_{c_1,c_2}(x_i) - y_i)\ (\sin(2\pi c_2 x) - 2\pi x c_2 \sin(2\pi c_1 x))$$
$$\frac{\partial E(c_1,c_2)}{\partial c_2} = \sum_{i=1}^{100} 2(g_{c_1,c_2}(x_i) - y_i)\ (2\pi c_1 x \cos(2\pi c_2 x) + \cos(2\pi c_1 x))$$

c) Evaluate and check that your gradient ∇E agrees at the following points:

$$\begin{pmatrix} 3 \\ 10 \end{pmatrix} \to \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 3.5 \\ 10 \end{pmatrix} \to \begin{pmatrix} 230 \\ 5.3 \end{pmatrix}, \begin{pmatrix} 4 \\ 10 \end{pmatrix} \to \begin{pmatrix} -100 \\ 9.8 \end{pmatrix}, \begin{pmatrix} 3 \\ 10.5 \end{pmatrix} \to \begin{pmatrix} 3.9 \\ 17.5 \end{pmatrix}, \begin{pmatrix} 3 \\ 11 \end{pmatrix} \to \begin{pmatrix} 4.7 \\ -7.6 \end{pmatrix}$$

d) Use `gradient_descent_nd()` to compute the optimal parameters $(\hat{c_1}, \hat{c_2})$ starting from the parameters $(c_1, c_2) = (3.1, 10.1)$.

e) To evaluate convergence properties, do a gradient descent starting at the following positions:

$$(c_1, c_2) \in \{(3.5, 10), (4, 10), (3, 10.5), (3, 11)\}$$

What are the final parameters for each start value ? Which are correct ?

Hint: Only two runs should converge to the optimum. The algorithm converges to local minima but not always to the correct solution. A local minimum is a position that is optimal for a small neighborhood, but not necessary globally.

f) To understand this behavior, it is useful to visualize the energy $E(c_1, c_2)$ in the range $(c_1, c_2) \in [0; 20] \times [0; 20]$. Create an image I and plot the energy levels (resolution of at least 300×300).

Hint: The framework provides a class `SimpleImage`. The listing below demonstrates how this class can be used:

```
#include "SimpleImage.h"

SimpleImage& vis = *SimpleImage::getInstance();
vis.create(300,300); //create an image with 300x300 pixels
vis(15,10) = 0.5; //set pixel (15,10) to value 0.5
vis.show(); //close window to continue
```

g) To differentiate values close to the optimum do the same plot again but apply a log transformation to the energies:

$$l(E(c_1, c_2)) \text{ with } l : x \to \log(x + 1)$$

Use this transformation for all further plots!

h) Plot another image storing the log-energy in the range $(c_1, c_2) \in [1; 5] \times [5; 15]$ (Resolution 100x100).

i) To see where points converge replot the last image. But instead of the energy at the starting point plot the final energy after convergence (log-scale!).

Hint: This requires a gradient descent for each pixel. This will be slow. Safe the result using a screenshot. For debugging you can change the termination bounds of gradient descent. You can enable the Release mode (Important, compiler settings are separate for debug/release).

Good luck !