Homework nr. 8

Consider $F : \mathbb{R}^n \longrightarrow \mathbb{R}$ a real function, $F(x) = F(x_1, x_2, \ldots, x_n)$. Approximate a (local or global) minimum point of the function F using the gradient descent method. Test the various methods for calculating the learning rate described in this text. Compute the gradient of the function F using both the analytical formula and the approximation Compare the solutions obtained using the two methods for computing the gradient of the function F in terms of the number of iterations required to reach the solutions (for the same precision $\epsilon > 0$). Test all the functions listed at the end of this text.

Bonus: Apply the minimization algorithms to solve the logistic regression problem described in the file RL (page 65).

You obtain **10pt** if you implement the already computed log-likelihood function and its gradient.

You obtain **15pt** if you build the log-likelihood function and its gradient using the data table.

$$\begin{aligned} \max\{F(x); x \in V\} &= -\min\{-F(x); x \in V\} \quad , \\ \operatorname{argmax}\{F(x); x \in V\} &= \operatorname{argmin}\{-F(x); x \in V\} \end{aligned}$$

Functions Minimization

Let $F : \mathbb{R}^n \longrightarrow \mathbb{R}$ be a real function, twice differentiable, $F \in C^2(\mathbb{R}^n)$, for which we want to approximate the solution x^* of the minimization problem:

$$\min\{F(x); x \in V\} \quad \longleftrightarrow \quad F(x^*) \leq F(x) \quad \forall (x, y) \in V \tag{1}$$

where $V = \mathbb{R}^n$ (x^* is a global minimum point) or $V = S(\bar{x}, r)$, is a sphere with the center $\bar{x}$ and the radius r (which is a local minimum point).

A point $\tilde{x}$ is a *critical point* for the function F, if it is a solution for the following system of equations:

$$\nabla F(\tilde{x}) = 0 \quad , \quad \nabla F(x) = \begin{pmatrix} \dfrac{\partial F}{\partial x_1}(x) \\ \vdots \\ \dfrac{\partial F}{\partial x_i}(x) \\ \vdots \\ \dfrac{\partial F}{\partial x_n}(x) \end{pmatrix} . \tag{2}$$

It is known that, for twice differentiable functions, the minimum points of the function F are found among the critical points. A critical point is a minimum if the Hessian matrix is positive semidefinite:

$$H(\tilde{x}) = \left(\frac{\partial^2 F}{\partial x_i \partial x_j}(\tilde{x}) \right)_{i,j=1,\ldots,n} \quad , \quad \big(H(\tilde{x})z, z\big)_{\mathbb{R}^n} \geq 0 \quad \forall z \in \mathbb{R}^n .$$

The Gradient Descendent Method

The minimum point of a function F is approximated by constructing a sequence of vectors $\{x^k\} \subseteq \mathbb{R}^n$ which, under certain conditions, converges to the desired minimum point x^*. The convergence of the sequence depends on the choice of the first element of the sequence x^0.

The $k+1$-element of the sequence x^{k+1}, is constructed from the previous one, x^k, as follows:

$$x^{k+1} = x^k - \eta_k \nabla F(x^k) \ , \ k = 0, 1, \ldots \ , \ x^0 - \text{ randomly selected} \tag{3}$$

The element η_k is called the learning rate, or the iteration step.

Strategies for Choosing the Learning Rate

1. $\eta_k = \eta$, $\forall k$ ($\eta = 10^{-3}, 10^{-4}, \ldots$). A constant learning rate with a too big value makes the minimum point hard to be found, while a too small value for the learning rate has the disadvantage of a too costing computation.

2. A possibility to solve problems with a constant learning rate is to consider a variable value, depending on the local context. The method described below is called *backtracking* adjustment of the step length/learning

rate (or *backtracking line search*). This method works for convex functions.

Consider $\beta \in (0, 1)$ a constant value (usually we take $\beta = 0.8$). At each step the learning rate is computed as follows:

$\eta = 1;$
$p = 1;$
while $F(x^k - \eta\nabla F(x^k)) > F(x^k) - \frac{\eta}{2}\|\nabla F(x^k)\|^2$ && $p < 8$
$\quad \eta = \eta\,\beta;$
$\quad p++;$

Important remark: The way in which the initial element, x^0 is chosen may cause the convergence or divergence of the sequence $\{x^k\}$ to the minimum point x^*. Usually, selecting the initial data in the vicinity of x^* assures the convergence $x^k \longrightarrow x^*$ for $k \to \infty$.

It is not necessary to store the entire sequence $\{x^k\}$; only the 'last' computed element x^{k_0}. We say that an element x^{k_0} approximates a minimum point, x^*, $x^{k_0} \approx x^*$ (where x^{k_0} is the last element of the sequence that is computed) when the difference between two successive elements of the sequence becomes sufficiently small, i.e.,

$$\|x^{k_0} - x^{k_0-1}\| \leq \epsilon \tag{4}$$

where ϵ is the precision with which we want to approximate the solution x^*.

Therefore, a possible approximation scheme of the solution x^* is the following:

Computation Scheme

randomly choose the initial values of the vector x;
$k = 0$;
do
{
- compute $\nabla F(x)$;
- compute the learning rate η using one of the two methods;
- $x = x - \eta \nabla F(x)$;
- $k = k + 1$;
}
while ($\eta \|\nabla F(x)\| \geq \epsilon$ and $k \leq k_{\max}$ and
$\eta \|\nabla F(x)\| \leq 10^{10}$)
if ($\eta \|\nabla F(x)\| \leq \epsilon$) $x \approx x^*$;
else *"divergence"* ; //(try to change the initial data)

A possible value for $k_{\max}$ is 30000 and $\epsilon > 10^{-5}$.

To compute the gradient of the function F in a certain point, the analytical gradient formula must be used (where the function is declared in the program). Also use the following approximation formula:

$$\nabla F(x) \approx \big(G_i(x,h)\big)_{i=1,\dots,n} \quad , \quad \frac{\partial F}{\partial x_i}(x) \approx G_i(x,h)$$

where

$$\frac{\partial F}{\partial x_i}(x) \approx G_i(x,h) = \frac{-F_1 + 8F_2 - 8F_3 + F_4}{12h} \ , \ \forall i = 1,\dots,n$$

with $h = 10^{-5}$ or 10^{-6} (may be considered as an input parameter), and:

$$\begin{array}{rcl} F_1 & = & F(x_1,\dots,x_{i-1},x_i+2h,x_{i+1},\dots,x_n), \\ F_2 & = & F(x_1,\dots,x_{i-1},x_i+h,x_{i+1},\dots,x_n), \\ F_3 & = & F(x_1,\dots,x_{i-1},x_i-h,x_{i+1},\dots,x_n), \\ F_4 & = & F(x_1,\dots,x_{i-1},x_i-2h,x_{i+1},\dots,x_n). \end{array}$$

Examples

$$l(w_0,w_1) = -\ln\left(1+\exp\left(w_0-w_1\right)\right) + w_0 + w_1 - \ln\left(1+\exp\left(w_0+w_1\right)\right)$$

$$\nabla l(w_0,w_1) = \begin{pmatrix} -\sigma(w_0-w_1) + \sigma(-w_0-w_1) \\ \sigma(w_0-w_1) + \sigma(-w_0-w_1) \end{pmatrix}, \quad \sigma(z) = \frac{1}{1+\exp(-z)}.$$

$$F(x_1,x_2) = x_1^2 + x_2^2 - 2x_1 - 4x_2 - 1, \nabla F(x_1,x_2) = \begin{pmatrix} 2x_1 - 2 \\ 2x_2 - 4 \end{pmatrix}, x_1^* = 1,\ \ x_2^* = 2.$$

$$F(x_1,x_2) = 3x_1^2 - 12x_1 + 2x_2^2 + 16x_2 - 10, \nabla F(x_1,x_2) = \begin{pmatrix} 6x_1 - 12 \\ 4x_2 + 16 \end{pmatrix}, x_1^* = 2, x_2^* = -4.$$

$$F(x_1,x_2) = x_1^2 - 4x_1x_2 + 5x_2^2 - 4x_2 + 3, \nabla F(x_1,x_2) = \begin{pmatrix} 2x_1 - 4x_2 \\ -4x_1 + 10x_2 - 4 \end{pmatrix}, x_1^* = 4, x_2^* = 2.$$

$$F(x_1,x_2) = x_1^2x_2 - 2x_1x_2^2 + 3x_1x_2 + 4, \nabla F(x_1,x_2) = \begin{pmatrix} 2x_1x_2 - 2x_2^2 + 3x_2 \\ x_1^2 - 4x_1x_2 + 3x_1 \end{pmatrix}, x_1^* = -1, x_2^* = 0.5.$$