Tehnici de Optimizare
Laborator 2
– Probleme fără constrângeri –

1 Metoda Gradient

Metoda Gradient reprezintă un algoritm de ordin I care, pornind dintr-un punct iniţial ales x^0, generează un şir de iteraţii (vectori) $x^1, x^2, \cdots$ pe baza direcţiei gradientului funcţiei obiectiv. Vom presupune că f are gradient $L-$continuu Lipschitz.

Metoda Gradient(x^0, ϵ)
Iniţializează $k = 0$.
Cât timp $\|\nabla f(x^k)\| \leq \epsilon$:
1. Calculează $\nabla f(x^k)$
2. Actualizează: $x^{k+1} = x^k - \alpha_k \nabla f(x^k)$
3. Set $k := k + 1$.

La linia 1, presupunem că un oracol de ordin I returnează gradientul ∇f evaluat în punctul curent x^k. Un aspect important al metodei este alegerea pasului $\alpha_k > 0$ dintre următoarele opţiuni:

- $\alpha_k = \alpha \in \left(0, \frac{2}{L}\right)$
- $\alpha_k = \arg\min_{\alpha \geq 0} f(x^k - \alpha \nabla f(x^k))$
- Alege $c > 0$, ajustează pasul α_k astfel încât să aibă loc relaţia de descreştere:

$$f(x^k - \alpha_k \nabla f(x^k)) \leq f(x^k) - c\alpha_k \|\nabla f(x^k)\|^2. \tag{1}$$

  Procedura de ajustare presupune alegerea $\rho \in (0, 1]$ şi actualizarea:

  (i) Alegem $c, \rho \in (0, 1), \alpha_{k,0} > 0$

  (ii) Cât timp $\alpha_{k,t}$ nu satisface (1) iterăm: $\alpha_{k,t+1} := \rho \alpha_{k,t}; \quad t := t + 1;$

1.1 Probleme propuse

1. Consideraţi problema pătratică:

$$\min_{x \in \mathbb{R}^n} \frac{1}{2}\|Ax - b\|^2 \ (=: f(x))$$

   a) Generaţi netrivial (A, b) astfel încât $\kappa = \frac{L}{\sigma} > 10^6$ şi $V^* > 10^3$ (verificaţi!)

   b) Pentru datele de la punctul a), implementaţi Metoda Gradient pentru fiecare opţiune a pasului α_k din secţiunea precedentă.

c) Implementaţi Metoda Gradient Stohastic prezentată la curs. Calculaţi limitele admise ale pasului α_k pentru care metoda converge la optim. Explicaţi aceste limite.

d) Scrieţi un algoritm iterativ: $x^{k+1} = x^k - \alpha_k s^k$ cu proprietăţile:

(i) s^k este ales aleator;

(ii) la fiecare iteraţie distanţa euclidiană dintre direcţia s^k şi $\nabla f(x^k)$ este în limitele $[0.01, 0.1]$;

(iii) algoritmul este convergent.

Explicaţi alegerile.

1.2 Ghid Python

Importare pachete necesare:

```
import numpy as np
from numpy import linalg
from numpy.linalg import norm
from numpy.linalg import eigvals
import matplotlib.pyplot as plt
```

Exemplu generare date:

```
n = 50
m = 70
A = np.random.randn(m,n)
b = np.random.randn(m,)
```

Exemplu calcul expresie gradient:

```
gradient = lambda x: np.matmul(A.T, np.matmul(A, x) - b)
```

Exemplu grafic curbe de convergenţă:

```
iterations_SG = range(100)
plt.xlabel("Iteratii")
plt.ylabel("f(x) - f*")
line = plt.semilogy(iterations_SG, f_traj_SG1, "b")
line = plt.semilogy(iterations_SG, f_traj_SG5, "r")
plt.legend(["SG1","SG2"])
plt.show()
```