統計と最適化　課題

陳 博洋　J4-230851

12/10/2024

(1)

$\min_{\epsilon_k>0} f(\mathbf{x}_k - \epsilon_k \nabla f(\mathbf{x}_k))$ について

$f(\mathbf{x}_k - \epsilon_k \nabla f(\mathbf{x}_k)) = \frac{1}{2}(\mathbf{x}_k - \epsilon_k \nabla f(\mathbf{x}_k))^\top \mathbf{A}(\mathbf{x}_k - \epsilon_k \nabla f(\mathbf{x}_k))$

これを ϵ_k に関して微分しゼロとおけば：

$\frac{\partial}{\partial \epsilon_k} f(\mathbf{x}_k - \epsilon_k \nabla f(\mathbf{x}_k)) = \frac{1}{2}(-\nabla f(\mathbf{x}_k))^\top \mathbf{A}(\mathbf{x}_k - \epsilon_k \nabla f(\mathbf{x}_k)) + \frac{1}{2}(\mathbf{x}_k - \epsilon_k \nabla f(\mathbf{x}_k))^\top \mathbf{A}(-\nabla f(\mathbf{x}_k)) = 0$

これを ϵ_k について解けば：

$$\epsilon_k = \frac{||\nabla f(\mathbf{x}_k)||^2}{\nabla f(\mathbf{x}_k)^\top \mathbf{A} \nabla f(\mathbf{x}_k)}$$

(2)

$\mathbf{A} = \begin{pmatrix} 20 & 0 \\ 0 & 2 \end{pmatrix}, \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ としておけば、$f(x_1, x_2) = 10x_1^2 + x_2^2 = \frac{1}{2}\mathbf{x}^\top \mathbf{A}\mathbf{x}$ となる.

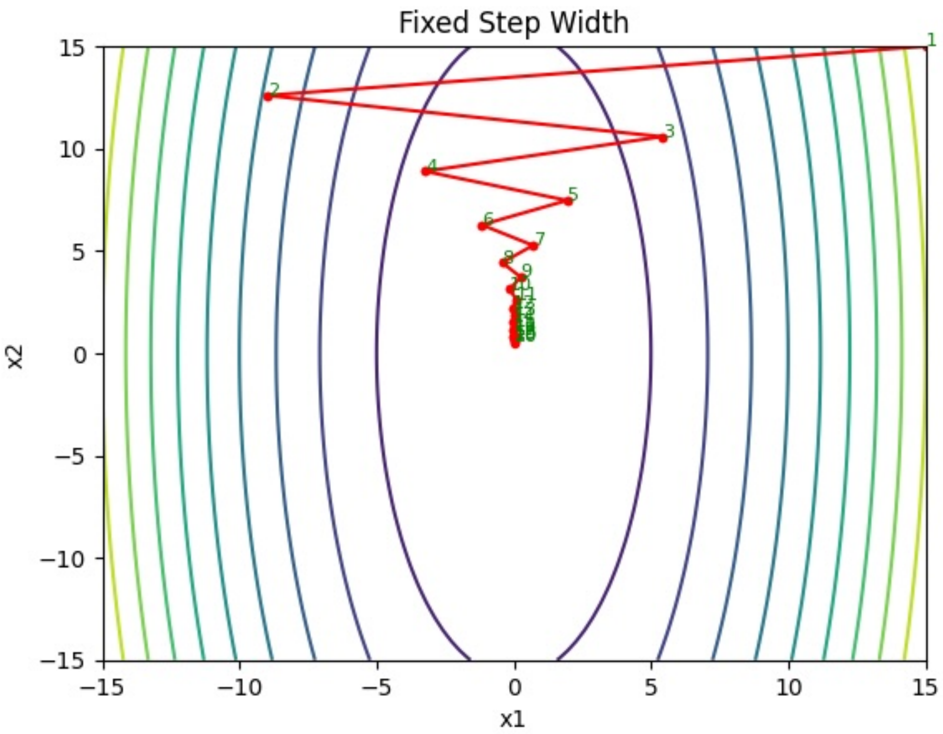
(1) の結果を利用すれば与えられた $f(x_1, x_2) = 10x_1^2 + x_2^2$ 厳密直線探索の解 ϵ_k が

$$\epsilon_k = \frac{||\nabla f(\mathbf{x}_k)||^2}{\nabla f(\mathbf{x}_k)^\top \mathbf{A} \nabla f(\mathbf{x}_k)} = \frac{100x_1^2 + x_2^2}{2000x_1^2 + 2x_2^2}$$

である。これを利用して以下のグラフを得られた。(設定、ソースコードは Appendix 参照)

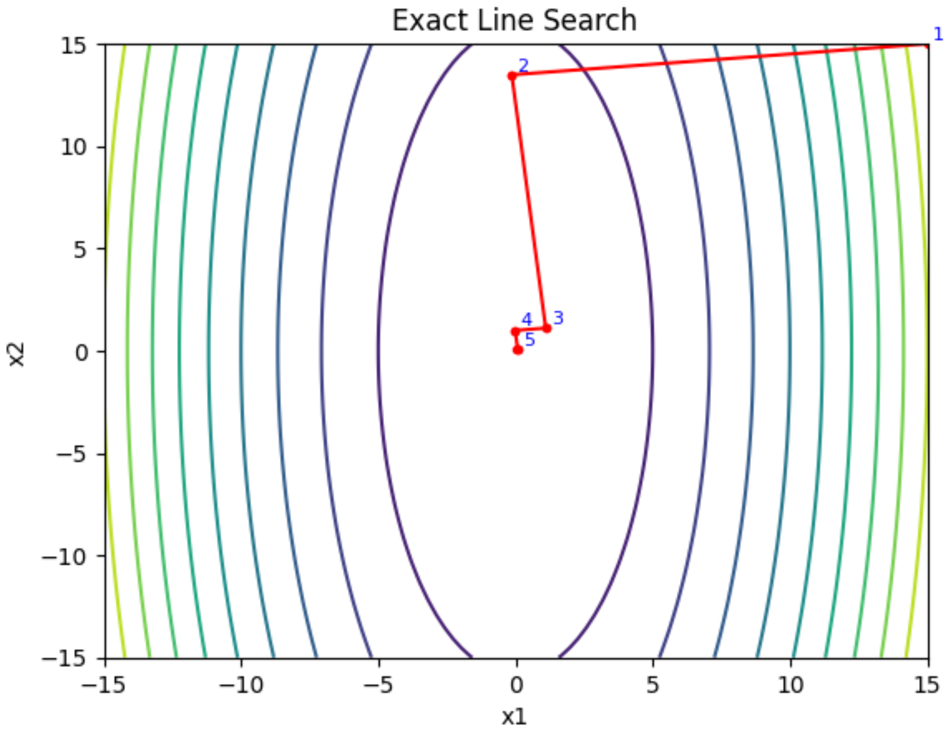
固定ステップ幅

図 1　固定ステップ幅を用いるときの探索様子



厳密直線探索

図 2　厳密直線探索を用いるときの探索様子



挙動の違い

厳密直線探索は予め計算された結果を用いて、効率的に停留点（目標）に近づくことができたが、固定ステップ幅探索はそのような”賢い”探索ができず、最終的に $x_1 = 0$ 周りに振動し、目標点に近づく速度が遅く、無駄な探索が多かった。

Appendix

探索の実装について幾つかの補足説明：

表 1　Parameter

探索の出発点	両方とも $(15, 15)$
固定ステップ幅 ϵ_k	0.08
収束の定義	$\|x1_k - x1_{k+1}\| < 0.1\| \wedge \|x2_k - x2_{k+1}\| < 0.1\|$

固定ステップ幅及び厳密直線探索それぞれのソースコード（元々は一つのグラフに統合したかったが、非常に見にくいため、別々にした、ただし、ソーズコードは ϵ_k の設定以外の部分はほぼ同じ）

Listing 1　固定ステップ幅

```
import numpy as np
import matplotlib.pyplot as plt

def f(x1, x2):
    return 10 * x1**2 + x2**2

def gradient_descent(x1, x2, e):
    x1_new = x1 - e * 20 * x1
    x2_new = x2 - e * 2 * x2
    return x1_new, x2_new

x1 = np.linspace(-15, 15, 100)
```

```python
x2 = np.linspace(-15, 15, 100)

X1, X2 = np.meshgrid(x1, x2)

Z = f(X1, X2)

CS = plt.contour(X1, X2, Z, 10)

plt.xlabel('x1')
plt.ylabel('x2')
plt.title('Fixed Step Width')

#初期値とイプシロン
x1_k, x2_k = 15, 15
e = 0.08

x1_history = [x1_k]
x2_history = [x2_k]

x1_k, x2_k = gradient_descent(x1_k, x2_k, e)

x1_history.append(x1_k)
x2_history.append(x2_k)

#収束条件を満たすまで繰り返す
while abs(x1_history[len(x1_history) - 1] - x1_history[len(x1_history) - 2]) > 0.1 or abs(x2_history[len(x2_history) - 1] - x2_history[len(x2_history) - 2]) > 0.1:
    x1_k, x2_k = gradient_descent(x1_k, x2_k, e)
    x1_history.append(x1_k)
    x2_history.append(x2_k)

for i in range(len(x1_history) - 1):
    plt.plot(x1_history[i:i+2], x2_history[i:i+2], 'ro-', markersize=3)
    plt.annotate(str(i + 1),
    xy=(x1_history[i], x2_history[i]),
    fontsize=8, color='green')

plt.show()
```

Listing 2　厳密直線探索

```python
import numpy as np
import matplotlib.pyplot as plt

def f(x1, x2):
    return 10 * x1**2 + x2**2

```

```
#をepsilon_kx_1,の関数として計算x_2
def gradient_descent(x1, x2):
    e = (100 * x1**2 + x2**2) / (2000 * x1**2 + 2 * x2**2)
    x1_new = x1 - e * 20 * x1
    x2_new = x2 - e * 2 * x2
    return x1_new, x2_new

x1 = np.linspace(-15, 15, 100)
x2 = np.linspace(-15, 15, 100)

X1, X2 = np.meshgrid(x1, x2)

Z = f(X1, X2)

CS = plt.contour(X1, X2, Z, 10)

plt.xlabel('x1')
plt.ylabel('x2')
plt.title('Exact Line Search')

#初期値
x1_k, x2_k = 15, 15

x1_history = [x1_k]
x2_history = [x2_k]

x1_k, x2_k = gradient_descent(x1_k, x2_k)

x1_history.append(x1_k)
x2_history.append(x2_k)

#収束条件を満たすまで繰り返す
while abs(x1_history[len(x1_history) - 1] - x1_history[len(x1_history) - 2]) > 0.1 or abs(x2_history[len(x2_history) - 1] - x2_history[len(x2_history) - 2]) > 0.1:
    x1_k, x2_k = gradient_descent(x1_k, x2_k)
    x1_history.append(x1_k)
    x2_history.append(x2_k)


for i in range(len(x1_history) - 1):
    plt.plot(x1_history[i:i+2], x2_history[i:i+2], 'ro-', markersize=3)
    plt.annotate(str(i + 1), xy=(x1_history[i], x2_history[i]),
                 xytext=(x1_history[i] + 0.2, x2_history[i] + 0.2),
                 fontsize=8, color='blue')

plt.show()
```