

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。

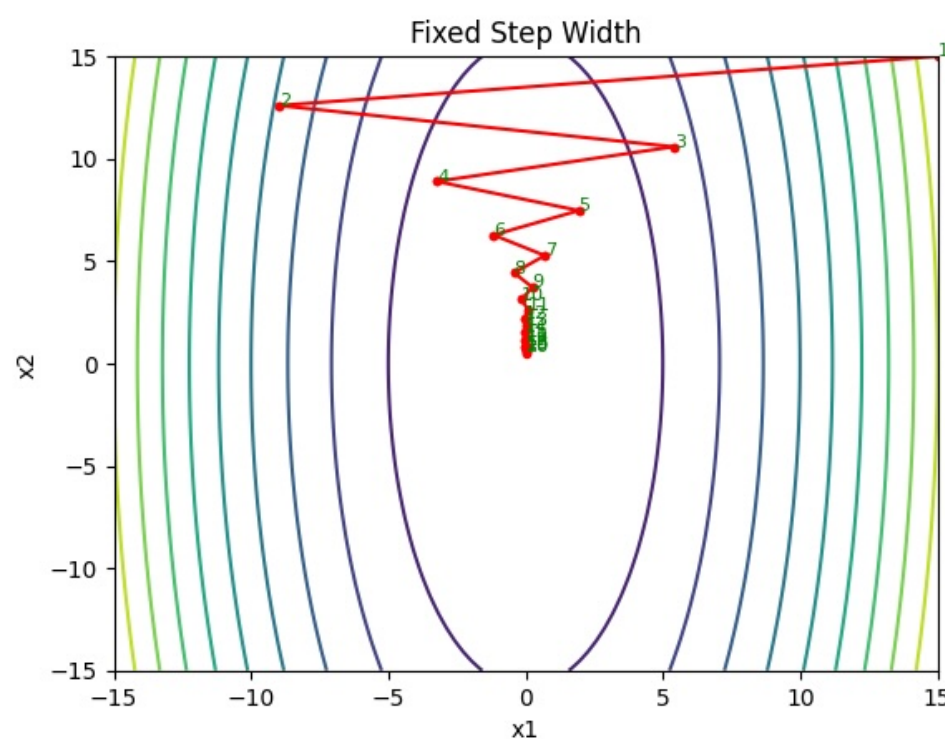
(1) の結果を利用すれば与えられた $f(x_1, x_2) = 10x_1^2 + x_2^2$ 厳密直線探索の解 ϵ_k が

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

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)

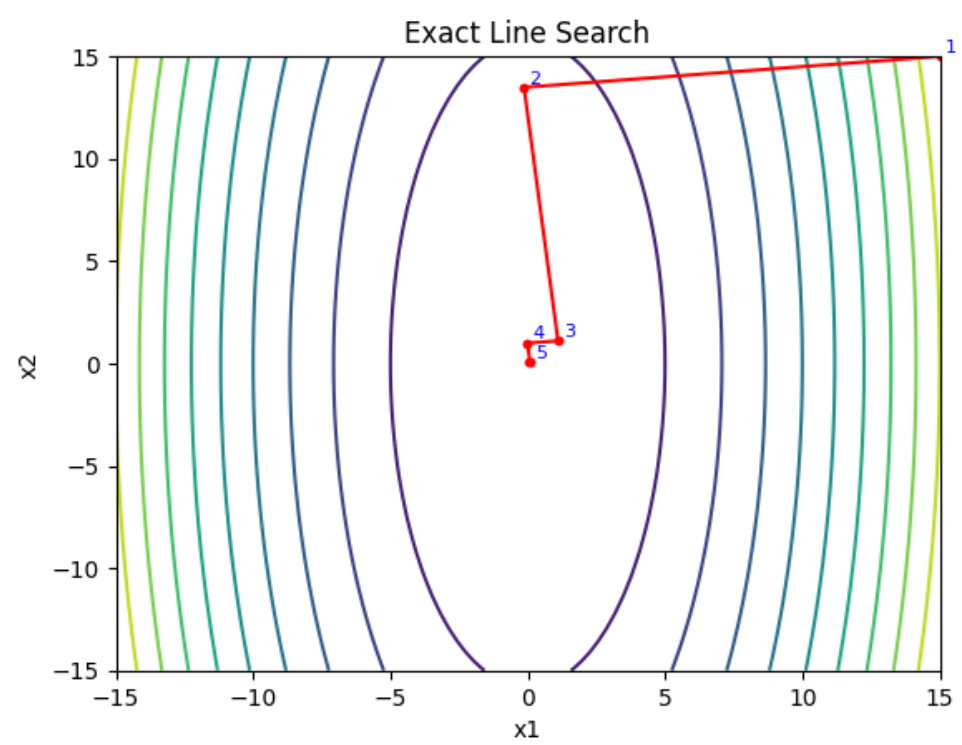
固定ステップ幅

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、ソースコードは ϵ_k の設定以外の部分はほぼ同じ）

Listing 1 固定ステップ幅

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def f(x1, x2):
5     return 10 * x1**2 + x2**2
6
7 def gradient_descent(x1, x2, e):
8     x1_new = x1 - e * 20 * x1
9     x2_new = x2 - e * 2 * x2
10    return x1_new, x2_new
11
12 x1 = np.linspace(-15, 15, 100)
```

```

13 x2 = np.linspace(-15, 15, 100)
14
15 X1, X2 = np.meshgrid(x1, x2)
16
17 Z = f(X1, X2)
18
19 CS = plt.contour(X1, X2, Z, 10)
20
21 plt.xlabel('x1')
22 plt.ylabel('x2')
23 plt.title('Fixed Step Width')
24
25 #初期値とイプシロン
26 x1_k, x2_k = 15, 15
27 e = 0.08
28
29 x1_history = [x1_k]
30 x2_history = [x2_k]
31
32 x1_k, x2_k = gradient_descent(x1_k, x2_k, e)
33
34 x1_history.append(x1_k)
35 x2_history.append(x2_k)
36
37 #収束条件を満たすまで繰り返す
38 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:
39     x1_k, x2_k = gradient_descent(x1_k, x2_k, e)
40     x1_history.append(x1_k)
41     x2_history.append(x2_k)
42
43 for i in range(len(x1_history) - 1):
44     plt.plot(x1_history[i:i+2], x2_history[i:i+2], 'ro-', markersize=3)
45     plt.annotate(str(i + 1),
46                 xy=(x1_history[i], x2_history[i]),
47                 fontsize=8, color='green')
48
49 plt.show()

```

Listing 2 厳密直線探索

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def f(x1, x2):
5     return 10 * x1**2 + x2**2
6

```

```

7  #をepsilon_kx_1,の関数として計算x_2
8  def gradient_descent(x1, x2):
9      e = (100 * x1**2 + x2**2) / (2000 * x1**2 + 2 * x2**2)
10     x1_new = x1 - e * 20 * x1
11     x2_new = x2 - e * 2 * x2
12     return x1_new, x2_new
13
14 x1 = np.linspace(-15, 15, 100)
15 x2 = np.linspace(-15, 15, 100)
16
17 X1, X2 = np.meshgrid(x1, x2)
18
19 Z = f(X1, X2)
20
21 CS = plt.contour(X1, X2, Z, 10)
22
23 plt.xlabel('x1')
24 plt.ylabel('x2')
25 plt.title('Exact Line Search')
26
27 #初期値
28 x1_k, x2_k = 15, 15
29
30 x1_history = [x1_k]
31 x2_history = [x2_k]
32
33 x1_k, x2_k = gradient_descent(x1_k, x2_k)
34
35 x1_history.append(x1_k)
36 x2_history.append(x2_k)
37
38 #収束条件を満たすまで繰り返す
39 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:
40     x1_k, x2_k = gradient_descent(x1_k, x2_k)
41     x1_history.append(x1_k)
42     x2_history.append(x2_k)
43
44
45 for i in range(len(x1_history) - 1):
46     plt.plot(x1_history[i:i+2], x2_history[i:i+2], 'ro-', markersize=3)
47     plt.annotate(str(i + 1), xy=(x1_history[i], x2_history[i]),
48         xytext=(x1_history[i] + 0.2, x2_history[i] + 0.2),
49         fontsize=8, color='blue')
50
51 plt.show()

```