最適化 Class-2 演習

21T2166D 渡辺大樹

2023 年 10 月 6 日

1　演習内容

Class-2 では探索における最適化を、関数の最大値を求めるアルゴリズムの設計を用いて理解していく。

今回用いるアルゴリズムは全探索と山登り探索になる。全探索は決められた定義域における関数の値をすべて求め、そこから最大の値となる関数の値と引数を特定する方法になる。

山登り探索はある初期値から関数の値が大きくなる方へ山に登るように移動し、最大となった点の関数の値と引数を返す方法になる。

この二つの探索を 1 変数関数と 2 変数関数で行っていく。実装したコードはソースコード 1,2,3,4 に示す。コードがかなり長いためレポート末に示す。

1.1　1 変数関数での探索

最大値を求める関数は (1)(2) の 2 種類を用いる。

$$f(x) = \sin(x) \tag{1}$$

$$g(x) = \frac{\sin(x^3 - 5(x+0.1)^2)}{x^3 + (x+0.1)^{-2}} \tag{2}$$

この二つの関数について、ある区間 $[x_{min}, x_{max}]$ においての最大値を求めるアルゴリズムを示していく。

全探索では探索する間隔となる $delta$ を用いて $[x_{min}, x_{max}]$ をこの $delta$ で分割したすべての x の値において $f(x)$ を計算する。この x での関数の出力値が $x - delta$ の出力値よりも大きい場合にのみ、その関数値と x の値をそれぞれ $fbest, xbest$ で保存する。

これにより区間すべての探索が終わった際、保存されていた $fbest, xbest$ がこの区間での関数の最大値となる。

山登り探索では探索する間隔 $delta$ と探索の初期値 x_0 を設定する。まずその探索にあたって探索の初期値から $\pm delta$ 分離れた関数の値と初期値での関数の値 $f(x_0 - delta), f(x_0), f(x_0 + delta)$ を用意する。この 3 つの関数の値を比較することで、グラフのいわば登るべき方向を決めることが

できる。$f(x_0 - delta)$ が大きければ x の負の方向に探索を進め、$f(x_0 + delta)$ が大きければ x の正の方向に探索を進める。$f(x_0)$ が大きいとその点が極大値となり、探索は終了する。ここで決めた方向は $sign$ に ± 1 の値で保存され、$delta$ に掛けてから x の値に足し算することで探索が進行する。

以降は $f(x)$ と $f(x + delta)$ の値を比較していき、$f(x + delta)$ が大きい場合は探索を続行し、$f(x)$ の値が大きくなる場合はその点が極大値となるため、探索はそこで終了する。

これにより探索終了時の x と $f(x)$ が出力される。

1.2 2 変数関数での探索

最大値を求める関数は (3)(4) の 2 種類を用いる。

$$f(x_1, x_2) = \frac{1}{2}(\sin(x_1) + \sin(x_2)) \tag{3}$$

$$g(x_1, x_2) = \frac{1}{2}\left(\frac{\sin({x_1}^3 - 5(x_1 + 0.1)^2)}{{x_1}^3 + (x_1 + 0.1)^{-2}} + \frac{\sin({x_2}^3 - 5(x_2 + 0.1)^2)}{{x_2}^3 + (x_2 + 0.1)^{-2}}\right) \tag{4}$$

2 変数関数であっても基本行うことは変わらない。そのためここでは相違点を示していく。最大値を求める区間は $[[x_{(1)min}, x_{(2)min}], [x_{(1)max}, x_{(2)max}]]$ となる。全探索ではこの区間すべてにおいて探索間隔 $delta$ を用いて関数値 $f(x_1, x_2)$ を計算し、最大値を探す。

山登り探索では探索の初期値 $(x_{(1)0}, x_{(2)0})$ と探索する間隔 $delta$ を同様に用意する。変更点は探索を始める際に決める探索の向きで、1 変数の際は x に対して正負どちらかの方向、もしくはその場が最大値となっていたが、2 変数となるので x_1, x_2 どちらに対しても $\pm delta$ したときの関数値がどうなるかを考えなければならないため、9 通りの進行方向が生まれる。

これを変更点としてソースコード 2 を 4 に変更している。

2 演習結果

以下に演習を行った結果を示す。

2.1 1 変数関数での探索

2.1.1 全探索

以下ではソースコード 1 を実行したときの結果を示す。

始めに関数として (1) 式を用い、探索の間隔を 0.01、探索範囲を $[0, \pi]$ としたときの結果が図 1 となる。

計算回数は 315 回で得られた最大値は $f(1.5700000000000012) = 0.9999996829318346$ となった。$\sin(x)$ は $\frac{\pi}{2}$ で最大値 1 を取るためおおよそ正確な値を得られている。

また探索範囲はそのままで探索の間隔を 0.001 と小さくすると計算回数 3142 回で得られた最大値は $f(1.5709999999999378) = 0.9999999792586128$ となった。理論値との誤差を比較するとお

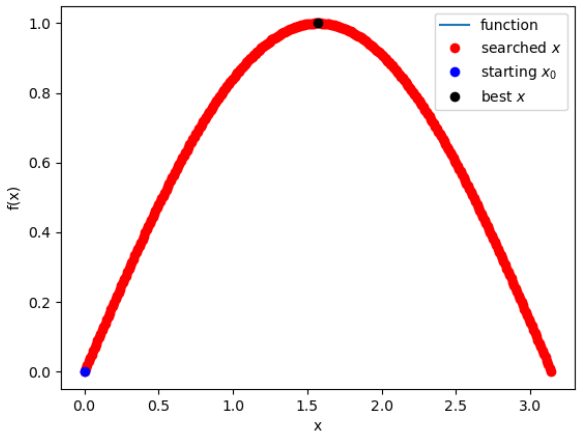

図 1　(1) 式を探索間隔 0.01, 探索範囲 $[0, \pi]$ で全探索を行った結果

およそ $\frac{1}{10}$ ほど小さくなっているが計算回数が 10 倍と多くなってしまっている。

続いて探索する関数として (2) 式を用いる。先ほどと同様

ソースコード 1　exhaustive_search.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Oct 5 17:36:56 2020
@author: hernan
"""
import matplotlib.pyplot as plt
import numpy as np

def sin(x):
    return np.sin(x)


def p3d(x):
    return np.sin(x**3 - 5*(x+0.1)**2) / (x**3 + (x+0.1)**-2)


def exhaustive_search(f, x_min, x_max, delta):

    txtname = "./text/exsearch_p3d_" + str(x_min) + "_" + str(x_max) + "_" +
         str(delta) + ".txt"
    fi = open(txtname, 'w')
    points = []
```

```
23      fvalues = []
24      x = x_min
25      fx = f(x)
26      fbest = fx
27      xbest = x
28      while x < x_max:
29          points.append(x)
30          fvalues.append(fx)
31
32          x = x + delta
33          fx = f(x)
34          if x <= x_max:
35              if fx > fbest:
36                  fbest = fx
37                  xbest = x
38          fi.write("f({}): {}\n".format(x, fx))
39
40      fi.write("best f({}): {}".format(xbest, fbest))
41      fi.close()
42
43      return points, fvalues, xbest, fbest
44
45  # 関数を選ぶ
46  #f = sin
47  f = p3d
48  # 変数の範囲を設定
49  x_min = -2
50  x_max = -1.1
51  # 探索の間隔を設定
52  delta = 0.001
53  # 全探索を呼び出す
54  x_search, fx_search, xbest, fbest = exhaustive_search(f, x_min, x_max, delta)
55  x_search = np.array(x_search)
56  fx_search = np.array(fx_search)
57  # 関数をプロットする
58  x_sample = np.arange(x_min, x_max, 0.001)
59  plt.plot(x_sample, f(x_sample))
60  # 探索点をプロットする
61  plt.plot(x_search, fx_search, 'ro')
62  plt.plot(x_search[0], fx_search[0], 'bo')
63  plt.plot(xbest, fbest, 'ko')
64  plt.xlabel("x")
```

```
65 plt.ylabel("f(x)")
66 #plt.legend(['関数','探索された x','探索の出発点','最高の探索点'])
67 plt.legend(['function', 'searched $x$', 'starting $x_0$', 'best $x$'])
68 dirname = "C://Program_Code/LaTeX/OPT/class2/"
69 filename = dirname + "exsearch_p3d_" + str(x_min) + "_" + str(x_max) + "_" +
       str(delta) + ".png"
70 plt.savefig(filename)
71 plt.show()
```

ソースコード 2　hill_climbing.py

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3
4
5  def sin(x):
6      return np.sin(x)
7
8
9  def p3d(x):
10     return np.sin(x**3 - 5*(x+0.1)**2) / (x**3 + (x+0.1)**-2)
11
12
13 def hill_climbing(f, x0, delta, x_min, x_max):
14     txtname = "./text/hillclimb_p3d_" + str(x_min) + "_" + str(x_max) + "_"
           + str(delta) + "_" + str(x0) + ".txt"
15     fi = open(txtname, 'w')
16     evaluations = np.array([f(x0-delta), f(x0), f(x0+delta)])
17     points = [x0]
18     fvalues = [evaluations[1]]
19
20     index_max = np.argmax(evaluations)
21
22     sign = 1.0
23     climb = True
24     if index_max == 0:
25         sign = -1.0
26     elif index_max == 2:
27         sign = 1.0
28     else:
29         climb = False
30
31     x = x0 + (sign*delta)
```

```
32      fx = evaluations[index_max]
33      fbest = fx
34      while climb == True:
35          points.append(x)
36          fvalues.append(fx)
37
38          x = x + (sign*delta)
39          fx = f(x)
40          if fx > fbest and (x <= x_max and x >= x_min):
41              fbest = fx
42              fi.write("f({}): {}\n".format(x, fx))
43          else:
44              climb = False
45
46      return points, fvalues
47
48
49  # 関数を選ぶ
50  # f = sin
51  f = p3d
52  # 変数の範囲を設定
53  x_min = -2
54  x_max = -1.1
55  # 探索の出発点
56  x0 = -1.3 # x_min < x0 < x_max
57  # 探索方向を決定するための間隔
58  delta = 0.001
59  # 山のりを呼び出す
60  points, fvalues = hill_climbing(f, x0, delta, x_min, x_max)
61  points = np.array(points)
62  fvalues = np.array(fvalues)
63  # 関数をプロットする
64  x = np.arange(x_min, x_max, 0.01)
65  plt.plot(x, f(x))
66  # 探索点をプロットする
67  plt.plot(points, fvalues, 'ro')
68  plt.plot(points[0], fvalues[0], 'bo')
69  plt.plot(points[len(points)-1], fvalues[len(fvalues)-1], 'ko')
70  plt.xlabel("x")
71  plt.ylabel("f(x)")
72  # plt.legend(['関数','探索された x','探索の出発点','最高の探索点'])
73  plt.legend(['function', 'searched $x$', 'starting $x_0$', 'best $x$'])
```

```
74 dirname = "C://Program_Code/LaTeX/OPT/class2/"
75 filename = dirname + "hillclimib_p3d_" + str(x_min) + "_" + str(x_max) + "_"
       + str(delta) + "_" + str(x0) + ".png"
76 plt.savefig(filename)
77 plt.show()
```

ソースコード 3　exhaustive_search_3d.py

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  """
4  Created on Mon Oct 5 17:36:56 2020
5  @author: hernan
6  """
7  import matplotlib.pyplot as plt
8  import numpy as np
9  import copy as cp
10
11 def sin_2v(x1,x2):
12     return (np.sin(x1) + np.sin(x2)) / 2
13
14
15 def p3d_2v(x1,x2):
16     return (np.sin(x1**3 - 5*(x1+0.1)**2) / (x1**3 + (x1+0.1)**-2) + np.sin
           (x2**3 - 5*(x2+0.1)**2) / (x2**3 + (x2+0.1)**-2)) / 2
17
18
19 def exhaustive_search(f, x_min, x_max, delta):
20
21     txtname = "./text/exsearch3d_sin_" + str(x_min) + "_" + str(x_max) + "_"
           + str(delta) + ".txt"
22     fi = open(txtname, 'w')
23     points = []
24     fvalues = []
25     x = cp.copy(x_min)
26     fx = f(*x)
27     fbest = fx
28     xbest = x
29     while x[0] < x_max[0]:
30         x[1] = x_min[1]
31         while x[1] < x_max[1]:
32             points.append(x)
33             fvalues.append(fx)
```

```
34
35              x[1] += delta
36              fx = f(*x)
37              if x <= x_max:
38                  if fx > fbest:
39                      fbest = fx
40                      xbest = cp.copy(x)
41              fi.write("f({}): {}\n".format(x, fx))
42          x[0] += delta
43
44      fi.write("best f({}): {}".format(xbest, fbest))
45      fi.close()
46
47      return points, fvalues, xbest, fbest
48
49  # 関数を選ぶ
50  f = sin_2v
51  # f = p3d_2v
52  # 変数の範囲を設定
53  x_min = [0,0]
54  x_max = [np.pi,np.pi]
55  # 探索の間隔を設定
56  delta = 0.01
57  # 全探索を呼び出す
58  x_search, fx_search, xbest, fbest = exhaustive_search(f, x_min, x_max, delta)
59  # x_search = np.array(x_search)
60  # fx_search = np.array(fx_search)
61  # # 関数をプロットする
62  # x_sample = np.arange(x_min, x_max, 0.001)
63  # plt.plot(x_sample, f(x_sample))
64  # # 探索点をプロットする
65  # plt.plot(x_search, fx_search, 'ro')
66  # plt.plot(x_search[0], fx_search[0], 'bo')
67  # plt.plot(xbest, fbest, 'ko')
68  # plt.xlabel("x")
69  # plt.ylabel("f(x)")
70  # #plt.legend(['関数','探索された x','探索の出発点','最高の探索点'])
71  # plt.legend(['function', 'searched $x$', 'starting $x_0$', 'best $x$'])
72  # dirname = "C://Program_Code/LaTeX/OPT/class2/"
73  # filename = dirname + "exsearch3d_p3d_" + str(x_min) + "_" + str(x_max) + "_
        " + str(delta) + ".png"
74  # plt.savefig(filename)
```

```
75 # plt.show()
```

ソースコード 4　hill_climbing_3d.py

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3  import copy as cp
4
5  def sin_2v(x1,x2):
6      return (np.sin(x1) + np.sin(x2)) / 2
7
8
9  def p3d_2v(x1,x2):
10     return (np.sin(x1**3 - 5*(x1+0.1)**2) / (x1**3 + (x1+0.1)**-2) + np.sin
           (x2**3 - 5*(x2+0.1)**2) / (x2**3 + (x2+0.1)**-2)) / 2
11
12
13 def hill_climbing(f, x0, delta, x_min, x_max):
14     txtname = "./text/hillclimb3d_sin_" + str(x_min) + "_" + str(x_max) + "_"
            + str(delta) + "_" + str(x0) + ".txt"
15     fi = open(txtname, 'w')
16     evaluations = np.array([
17         [f(x0[0]-delta,x0[1]-delta), f(x0[0]-delta,x0[1]), f(x0[0]-delta,x0
               [1]+delta)],
18         [f(x0[0],x0[1]-delta), f(*x0), f(x0[0],x0[1]+delta)],
19         [f(x0[0]+delta,x0[1]-delta), f(x0[0]+delta,x0[1]), f(x0[0]+delta,x0
               [1]+delta)]
20         ])
21     points = [x0]
22     fvalues = [evaluations[1]]
23
24     index_max = np.unravel_index(np.argmax(evaluations), evaluations.shape)
25
26     sign = np.array([1.0,1.0]) #登る方向
27     climb = True
28     if index_max[0] == 0:
29         sign[0] = -1.0
30         if index_max[1] == 0:
31             sign[1] = -1.0
32         elif index_max[1] == 2:
33             sign[1] = 1.0
34         else:
35             sign[1] = 0.0
```

```
36      elif index_max[0] == 2:
37          sign[0] = 1.0
38          if index_max[1] == 0:
39              sign[1] = -1.0
40          elif index_max[1] == 2:
41              sign[1] = 1.0
42          else:
43              sign[1] = 0.0
44      else:
45          sign[0] = 0.0
46          if index_max[1] == 0:
47              sign[1] = -1.0
48          elif index_max[1] == 2:
49              sign[1] = 1.0
50          else:
51              climb = False
52
53
54      x = x0 + np.array([sign[0]*delta,sign[1]*delta])
55      fx = evaluations[index_max]
56      fbest = fx
57      while climb == True:
58          points.append(x)
59          fvalues.append(fx)
60
61          x = x + np.array([sign[0]*delta,sign[1]*delta])
62          fx = f(*x)
63          if fx > fbest and (all(x <= x_max) and all(x >= x_min)):
64              fbest = fx
65              fi.write("f({}): {}\n".format(x, fx))
66          else:
67              climb = False
68
69      return points, fvalues
70
71
72  # 関数を選ぶ
73  f = sin_2v
74  # f = p3d_2v
75  # 変数の範囲を設定
76  x_min = np.array([0,0])
77  x_max = np.array([np.pi, np.pi])
```

```
78 # 探索の出発点
79 x0 = np.array([0.6,0.6]) # x_min < x0 < x_max
80 # 探索方向を決定するための間隔
81 delta = 0.05
82 # 山のりを呼び出す
83 points, fvalues = hill_climbing(f, x0, delta, x_min, x_max)
84 # points = np.array(points)
85 # fvalues = np.array(fvalues)
86 # # 関数をプロットする
87 # x = np.arange(x_min, x_max, 0.01)
88 # plt.plot(x, f(x))
89 # # 探索点をプロットする
90 # plt.plot(points, fvalues, 'ro')
91 # plt.plot(points[0], fvalues[0], 'bo')
92 # plt.plot(points[len(points)-1], fvalues[len(fvalues)-1], 'ko')
93 # plt.xlabel("x")
94 # plt.ylabel("f(x)")
95 # # plt.legend(['関数','探索された x','探索の出発点','最高の探索点'])
96 # plt.legend(['function', 'searched $x$', 'starting $x_0$', 'best $x$'])
97 # dirname = "C://Program_Code/LaTeX/OPT/class2/"
98 # filename = dirname + "hillclimib_p3d_" + str(x_min) + "_" + str(x_max) + "_
       " + str(delta) + "_" + str(x0) + ".png"
99 # plt.savefig(filename)
100 # plt.show()
```