Python小练习: 优化器torch.optim的使用

作者: 凯鲁嘎吉 - 博客园 http://www.cnblogs.com/kailugaji/

本文主要介绍Pytorch中优化器的使用方法,了解optimizer.zero_grad()、loss.backward()以及optimizer.step()函数的用法。

问题陈述:假设最小化目标函数为\$L = \sum\nolimits_{i = 1}^N {x_i^2} \$。给定初始值\$\left[{x_1^{(0)},x_2^{(0)}} \right] = [5,{\rm{}}10]\$, 求最优解\$\hat x = \arg \min L\$。

1. optim_test.py

```
1 # -*- coding: utf-8 -*-
2 # Author: 凯鲁嘎吉 Coral Gajic
3 # https://www.cnblogs.com/kailugaji/
4 # Python小练习: 优化器torch.optim的使用
5 # 假设损失函数为loss = \Sigma(x^2),给定初始x(0),目的是找到最优的x,使得loss最小化
6 # 以2维数据为例, loss = x1<sup>2</sup> + x2<sup>2</sup>
7 # loss对x1的偏导为2 * x1, loss对x2的偏导为2 * x2
9 部分参考:
      https://www.cnblogs.com/zhouyang209117/p/16048331.html
12 import torch
13 import torch. optim as optim
14 import numpy as np
15 import matplotlib.pyplot as plt
16 plt.rc('font', family='Times New Roman')
17
18 # 方法一:
19 # 自己实现的优化
20 rate = 0.1 # 学习率
21 iteration = 30 # 迭代次数
22 \text{ num} = 3
23 data = np. array([5.0, 10.0]) # 给定初始数据x(0)
24 for i in range (iteration):
      loss = (data ** 2). sum(axis = 0) # 优化目标: 最小化loss = x1^2 + x2^2
      my_grad = 2 * data # 梯度d(loss)/dx: 2 * x1, 2 * x2
26
27
      print('%d.' %(i+1),
            '数据:', np. around (data, num),
28
            '\t损失函数:', np. around (loss, num),
29
```

```
'\t梯度:', np. around (my grad, num)
30
31
      data -= my grad * rate # 优化更新: x = x - learning rate * (d(loss)/dx)
32
33
34 print('-----')
35 # 方法二:
36 # pytorch自带的优化器
37 data = np. array([5.0, 10.0]) # 给定初始数据x(0)
38 data = torch.tensor(data, requires grad=True) # 需要求梯度
39 optimizer = optim. SGD([data], 1r = rate) # 优化器: 随机梯度下降
40 plot loss = []
41 for i in range (iteration):
      optimizer.zero grad() # 清空先前的梯度
42
43
      loss = (data ** 2).sum() # 优化目标: 最小化loss = x1^2 + x2^2
44
      print('%d.' %(i+1),
           '数据:', np.around(data.detach().numpy(), num),
45
46
           '\t损失函数:', np. around(loss. item(), num),
           end=' '
47
48
49
      loss.backward() # 计算当前梯度d(loss)/dx = 2 * x, 并反向传播
      print('\t梯度: ', np.around(data.grad.detach().numpy(), num)) # 打印梯度
50
      optimizer.step() # 优化更新: x = x - learning rate * (d(loss)/dx)
51
52
      plot loss.append([i+1, loss.item()]) # 保存每次迭代的loss
53
54 plot loss = np. array (plot loss) # 将list转换成numpy
56 plt.plot(plot loss[:, 0], plot loss[:, 1], color = 'red', ls = '-')
57 plt. xlabel ('Iteration')
58 plt. ylabel ('Loss')
59 plt. tight layout()
60 plt. savefig ('Loss.png', bbox inches='tight', dpi=500)
61 plt. show()
```

2. 结果

```
D:\ProgramData\Anaconda3\python.exe "D:\Python code/2023.3 exercise/optimizer/optim test.py"
1. 数据: [5.10.]
                    损失函数: 125.0
                                     梯度: [10. 20.]
2. 数据: [4. 8.]
                  损失函数: 80.0
                                  梯度: [8.16.]
3. 数据: [3.2 6.4]
                   损失函数: 51.2
                                    梯度: [6.4 12.8]
4. 数据: [2.56 5.12]
                     损失函数: 32.768
                                        梯度: [5.12 10.24]
  数据: [2.048 4.096]
                       损失函数: 20.972
                                         梯度: [4.096 8.192]
  数据: [1.638 3.277]
                       损失函数: 13.422
                                         梯度: [3.277 6.554]
                       损失函数: 8.59
7. 数据: [1.311 2.621]
                                        梯度: [2.621 5.243]
                       损失函数: 5.498
                                        梯度: [2.097 4.194]
  数据: [1.049 2.097]
9. 数据: [0.839 1.678]
                       损失函数: 3.518
                                         梯度: [1.678 3.355]
```

```
10. 数据:
          [0.671 1.342]
                          损失函数: 2.252
                                             梯度: [1.342 2.684]
11. 数据:
                                                  [1.074 2.147]
          [0.537 \ 1.074]
                                             梯度:
                          损失函数: 1.441
12. 数据:
          [0.429 \ 0.859]
                          损失函数: 0.922
                                             梯度: [0.859 1.718]
                          损失函数: 0.59
13. 数据:
          [0.344 \ 0.687]
                                            梯度: [0.687 1.374]
14. 数据:
                          损失函数: 0.378
                                             梯度: [0.55 1.1]
          [0.275 \ 0.55]
15. 数据:
          [0.22 \ 0.44]
                        损失函数: 0.242
                                           梯度: [0.44 0.88]
16. 数据:
                          损失函数: 0.155
          [0.176 \ 0.352]
                                             梯度: [0.352 0.704]
17. 数据:
                          损失函数: 0.099
          [0.141 \ 0.281]
                                             梯度: [0.281 0.563]
   数据:
                          损失函数: 0.063
          [0.113 \ 0.225]
                                             梯度: [0.225 0.45]
18.
19. 数据:
                        损失函数: 0.041
          [0.09 \ 0.18]
                                           梯度: [0.18 0.36]
20. 数据:
                          损失函数: 0.026
          [0.072 \ 0.144]
                                             梯度: [0.144 0.288]
                          损失函数: 0.017
21. 数据:
          [0.058 \ 0.115]
                                             梯度: [0.115 0.231]
22. 数据:
                          损失函数: 0.011
          [0.046 \ 0.092]
                                             梯度: [0.092 0.184]
23. 数据:
          [0.037 \ 0.074]
                          损失函数: 0.007
                                             梯度: [0.074 0.148]
   数据:
                          损失函数: 0.004
                                             梯度:
          [0.03 \quad 0.059]
                                                   [0.059 \ 0.118]
24.
25. 数据:
                          损失函数: 0.003
                                             梯度:
          [0.024 \ 0.047]
                                                   [0.047 \ 0.094]
26. 数据:
                          损失函数: 0.002
          [0.019 \ 0.038]
                                             梯度:
                                                   [0.038 \ 0.076]
27. 数据:
                          损失函数: 0.001
                                             梯度: [0.03 0.06]
          [0.015 \ 0.03]
                          损失函数: 0.001
28. 数据:
          [0.012 \ 0.024]
                                             梯度: [0.024 0.048]
                                           梯度: [0.019 0.039]
29. 数据:
          [0.01 \ 0.019]
                          损失函数: 0.0
30. 数据:
         [0.008 0.015]
                          损失函数: 0.0
                                           梯度: [0.015 0.031]
                                         梯度: [10. 20.]
1. 数据:
                     损失函数: 125.0
         [ 5. 10.]
2. 数据:
         [4. 8.]
                    损失函数: 80.0
                                      梯度: [8.16.]
                                       梯度:
3. 数据:
         [3.26.4]
                     损失函数: 51.2
                                             [6.4 12.8]
                       损失函数: 32.768
  数据:
         [2.565.12]
                                           梯度: [5.12 10.24]
4.
  数据:
                         损失函数: 20.972
                                             梯度: [4.096 8.192]
         [2.048 4.096]
  数据:
                         损失函数: 13.422
                                             梯度: [3.277 6.554]
         [1.638 \ 3.277]
  数据:
                         损失函数: 8.59
         [1.311 2.621]
                                           梯度: [2.621 5.243]
7.
8. 数据:
         [1.049 2.097]
                         损失函数: 5.498
                                            梯度: [2.097 4.194]
9. 数据:
                         损失函数: 3.518
         [0.839 \ 1.678]
                                            梯度: [1.678 3.355]
10. 数据:
          [0.671 1.342]
                          损失函数: 2.252
                                             梯度: [1.342 2.684]
11. 数据:
                                             梯度: [1.074 2.147]
          [0.537 \ 1.074]
                          损失函数: 1.441
12. 数据:
                          损失函数: 0.922
                                             梯度: [0.859 1.718]
          [0.429 \ 0.859]
13. 数据:
          [0.344 \ 0.687]
                          损失函数: 0.59
                                            梯度: [0.687 1.374]
14. 数据:
                          损失函数: 0.378
                                             梯度: [0.55 1.1]
          [0.275 \ 0.55]
15. 数据:
          [0.22 \ 0.44]
                        损失函数: 0.242
                                           梯度: [0.44 0.88]
16. 数据:
          [0.176 \ 0.352]
                          损失函数: 0.155
                                             梯度: [0.352 0.704]
17. 数据:
                                             梯度: [0.281 0.563]
          [0.141 0.281]
                          损失函数: 0.099
18. 数据:
                          损失函数: 0.063
          [0.113 \ 0.225]
                                             梯度: [0.225 0.45]
19. 数据:
          [0.09 \ 0.18]
                        损失函数: 0.041
                                           梯度: [0.18 0.36]
                                             梯度: [0.144 0.288]
                          损失函数: 0.026
20. 数据:
          [0.072 \ 0.144]
                          损失函数: 0.017
21. 数据:
          [0.058 \ 0.115]
                                             梯度: [0.115 0.231]
22. 数据:
                          损失函数: 0.011
          [0.046 \ 0.092]
                                             梯度: [0.092 0.184]
                          损失函数: 0.007
23.
   数据:
          [0.037 \ 0.074]
                                             梯度: [0.074 0.148]
   数据:
                          损失函数: 0.004
24.
          [0.03 \quad 0.059]
                                             梯度:
                                                   [0.059 \ 0.118]
25. 数据:
                          损失函数: 0.003
                                             梯度: [0.047 0.094]
         [0.024 \ 0.047]
```

26. 数据: [0.019 0.038] 损失函数: 0.002 梯度: [0.038 0.076] 损失函数: 0.001 损失函数: 0.001 损失函数: 0.0 损失函数: 0.0 27. 数据: [0.015 0.03] 梯度: [0.03 0.06] 28. 数据: 梯度: [0.024 0.048] [0.012 0.024] 29. 数据: $[0.01 \quad 0.019]$ 梯度: [0.019 0.039] 30. 数据: [0.008 0.015] 梯度: [0.015 0.031]

Process finished with exit code 0

