《人工智能导论》

实验四：模拟退火算法

学　号 22920212204396

姓　名 黄子安

2024 年 5 月 2 日

实验四：模拟退火算法

229202212204396 黄子安

一、实验目的

模拟退火算法(Simulated Annealing，SA)最早的思想是由 N. Metropolis 等人于 1953 年提出。1983 年,S.Kirkpatrick 等成功地将退火思想引入到组合优化领域。它是基于 Monte-Carlo 迭代求解策略的一种随机寻优算法，其出发点是基于物理中固体物质的退火过程与一般组合优化问题之间的相似性。本实验通过解决一个一元函数的最值问题，更好的熟悉和掌握模拟退火算法。

二、实验内容

利用模拟退火算法寻找$f(x) = 11sin(6x) + 7cos(5x), x \in [0,2\pi]$的最小值

三、实验过程

先定义一下题目中所有求最值的函数$f(x)$

```
def f(x):
    return 11 * np.sin(6 * x) + 7 * np.cos(5 * x)
```

之后是退火算法的主体部分，结合下面流程图进行解释

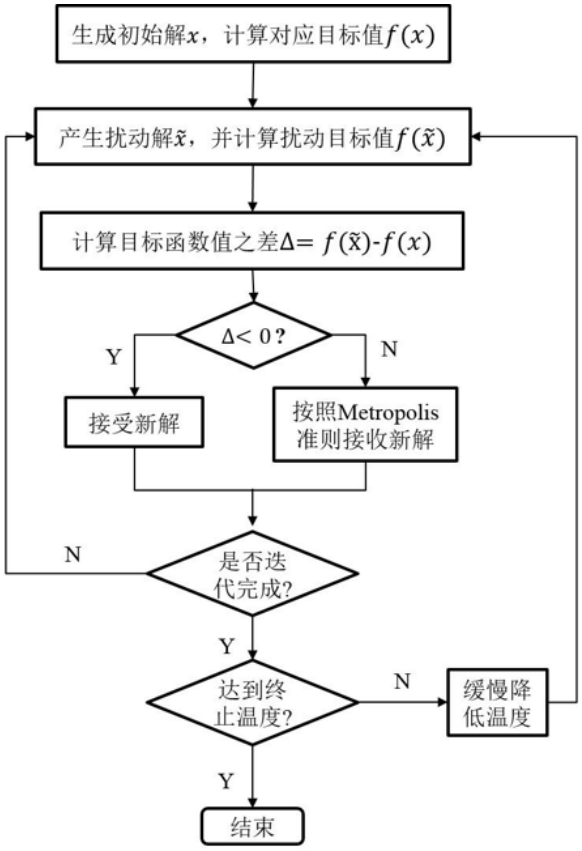
```
def simulated_annealing(f, x_min, x_max, T_max, T_min, cooling_rate, max_iter, epsilon):
    x_best = 3.5
    f_best = f(x_best)

    x_current = x_best
    f_current = f_best

    x_hist = [x_best]
    f_hist = [f_best]

    T = T_max
    while T > T_min:
        for _ in range(max_iter):
            x_new = x_current + np.random.uniform(-epsilon, epsilon)
            x_new = max(min(x_new, x_max), x_min)  # 确保 x_new 在 [x_min, x_max] 范围内
            f_new = f(x_new)
            delta_f = f_new - f_current
            if delta_f < 0 or np.random.rand() < np.exp(-delta_f / T):
                x_current = x_new
                f_current = f_new
                if f_current < f_best:
                    x_best = x_current
                    f_best = f_current
            x_hist.append(x_best)
            f_hist.append(f_best)
        T *= cooling_rate
    return x_best, f_best, x_hist, f_hist
```

1、首先给定一个初始解，计算对应的函数值，这里为了让迭代过程更加直观些，初值选择 3.5

2、对于当前温度T，进行马尔科夫链迭代，对当前的x值进行微扰动，生成一个附近邻域的新解，之后判断这个新的解对应的函数值是否比原来的值更小，如果更小则直接接受这个新的解作为新一轮迭代中的x，否则根据 Metropolis 准则按照一定的概率接受这个新的解，接受当前这个解的概率为$e^{-\frac{fnew-f}{T}}$，这里指数本来还有一个系数，但是对于编程求解影响不大直接设置为 1

3、完成一轮马尔科夫链迭代后降低当前体系的温度，温度的衰减使用指数衰减，通过累乘一个系数实现，这个系数需要比较小于并且接近于 1，从而保证体系的降温过程有一定的时间，保证退火的有效性

4、如果体系的温度小于给定要求的最低温度，则退火结束，输出当前的最优解



编写方法调用退火算法进行，输出最优解，对应的冷却参数表如代码所示

```python
if __name__ == '__main__':

    # 冷却参数表
    x_min = 0
    x_max = 2 * np.pi
    T_max = 100
    T_min = 0.1
    cooling_rate = 0.95
    max_iter = 100
    epsilon = 0.1  # 马尔科夫迭代步长

    x_best, f_best, x_hist, f_hist = simulated_annealing(f, x_min, x_max,
                                                         T_max, T_min, cooling_rate,
                                                         max_iter, epsilon)

    print("最优解 x:", x_best)
    print("最小值 f(x):", f_best)
```

python

在退火的过程中记录了历史搜索过的点，最后将搜索过程进行可视化

```python
    plt.figure(figsize=(10, 6))
    plt.plot(np.linspace(x_min, x_max, 1000)
             , f(np.linspace(x_min, x_max, 1000)), 'r-', label='f(x)')
    plt.plot(x_hist, f_hist, 'bo-', markersize=2, label='Search Path')
    plt.scatter(x_hist[0], f_hist[0], color='green', label='Start Point')
    plt.scatter(x_best, f_best, color='red', label='End Point')

    plt.xlabel('x')
    plt.ylabel('f(x)')
    plt.title('Simulated Annealing Search for Minimum of f(x)')
    plt.legend()
    plt.grid(True)
    plt.show()
```

三、实验结果

函数的最值和最值点如下图所示

```
D:\anaconda3\python.exe D:\Desktop\learning\3.2\人工智能导论\lab\lab4\code.py
最优解 x: 1.8495607506032825
最小值 f(x): -17.833728584698676

Process finished with exit code 0
```

对应的搜索路径如下图所示，绿色点为起点，红色点为终点，可以看到一开始向右搜索，继续保持下去将会导致陷入局部最优解，这个时候因为退火算法的作用选择了一个可能不是很好的解，跳转到了左侧

之后继续下降，到达另一个局部最优解，这个时候函数的差值越来越小，根据 Metropolis 准则，跳转的概率越来越大，最后在几次迭代后再一次跳转

到达最优解所在的谷底时，继续开始迭代下降，虽然根据 Metropolis 准则，函数差值减小会提高跳转概率，但是随着体系温度降低，跳转的概率会减小，在两者综合的作用下还是体系温度下降的影响更大，最后趋于稳定，当温度够低的时候退火结束，输出最优解