# 无约束的最优化问题

本项目实现实现了一个无约束的最优化问题，目标是求解函数 $f(x_1, x_2)$ 的最小值

总体来说，这段代码实现了一个基本的梯度下降算法，用于求解无约束的最优化问题。

## 运行指南

基于 Python 标准库，无需下载任何第三方库。对 Python 版本等均无任何要求。

### 运行测试

```
cd code2
python main.py
```

## 运行结果

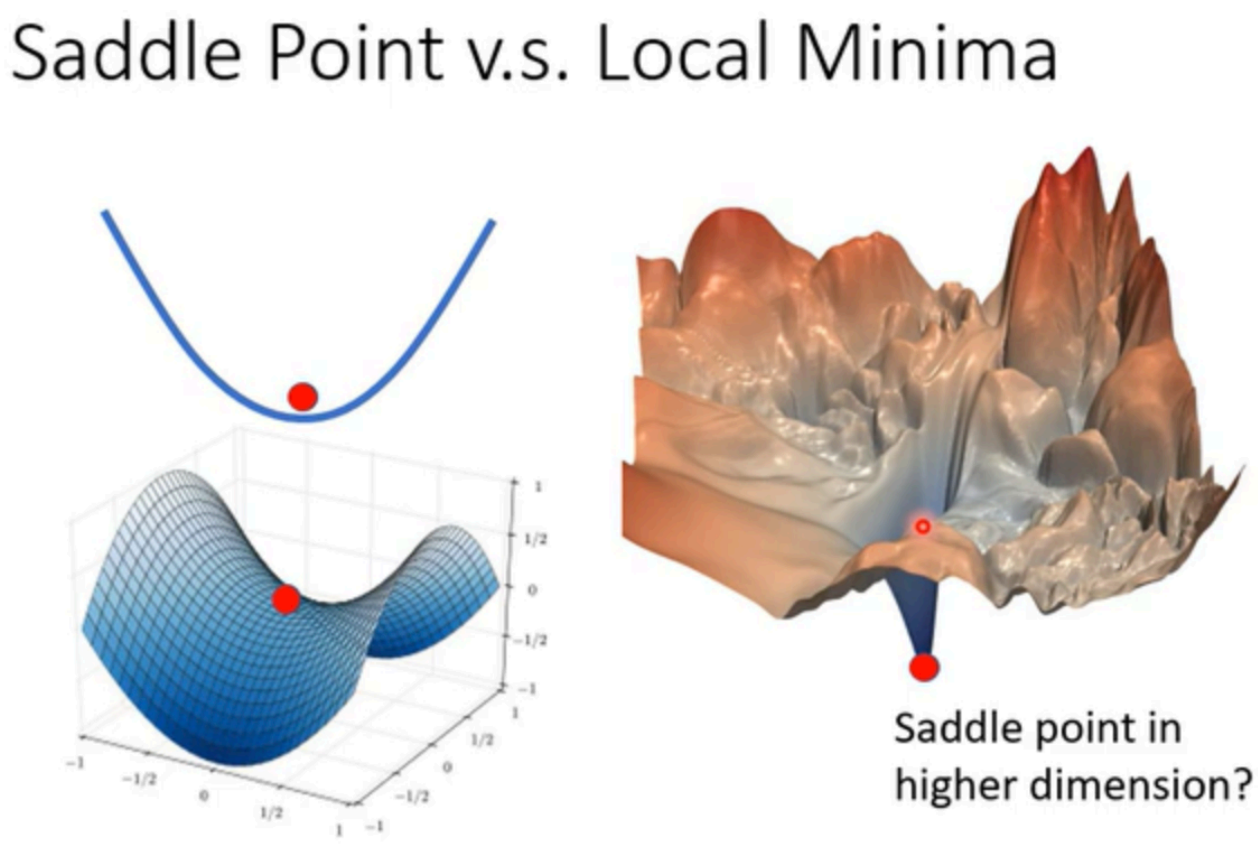
```
❯ python main.py
x1:  -0.34657359027958445 x2:  0.0
f(x1, x2):  2.5592666966582156
```

## 思考内容

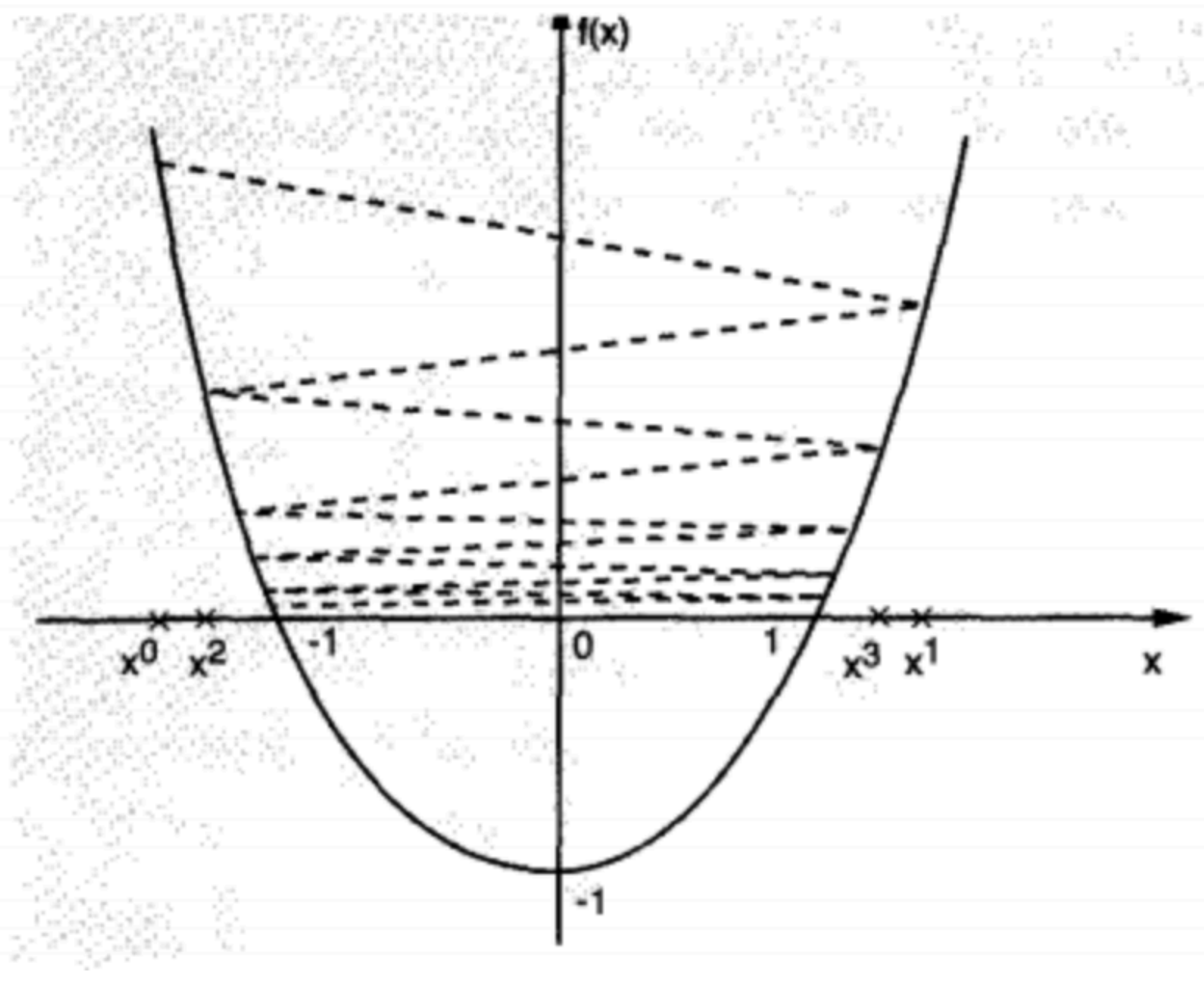
> 对于凸函数而言，使用迭代方法最后会收敛到最小值点，而对于非凸函数，一定会收敛到最小值点吗？初始点的设置对于非凸函数极值点的求解有多大影响？一般有哪些初始点设置方法？步长又有多少影响？步长要怎样设置？梯度下降法的缺点有哪些？针对这些缺点，有哪些改进的迭代方法？

对于非凸函数，不一定收敛到最小值点，但通常会收敛到局部最小值点或鞍点。收敛到局部最小值点时，梯度等于 0，无法再继续更新，此时为局部最优解；收敛到鞍点时，梯度同样等于 0，但此时既不是全局最优解，也不是局部最优解，如下图中所示，鞍点位置的四周有比鞍点值小的地方，但由于梯度为 0，无法再进行更新，因此，鞍点是我们不希望出现的情况。

初始点的设置对于非凸函数极值点的求解有很大影响，不同初始点可能会导致不同的收敛结果。初始点选择不好，可能会陷入到局部最小值点或鞍点。可以采用随机初始化的方式。




步长同样也有影响，选取过小会导致收敛速度太慢，选取过大可能会导致无法收敛，如下图，在最小值点来回移动，而始终到达不了最小值点。可以采用自适应的方法来动态调整步长。



梯度下降法的缺点：

1. 可能会收敛到局部最小值点或鞍点，而不是全局最小值点。
2. 对于非凸函数，需要多次尝试不同的初始点，才能找到最优解。
3. 对于高维问题，计算梯度的代价很高，可能会导致算法收敛速度很慢。
4. 对于非凸函数，可能需要使用更复杂的优化算法，如牛顿法、拟牛顿法等。

改进：

1. 采用随机梯度下降法 SGD：每次用一个小样本来计算梯度。
2. 动量梯度下降法：引入动量项来加速收敛速度，减少因步长太大而来回震荡的现象。
3. 自适应学习率方法：如 Adam，动态调整学习率，即步长。