最优化方法

课程报告

姓名：　　朱培懿

学号：　2023311810

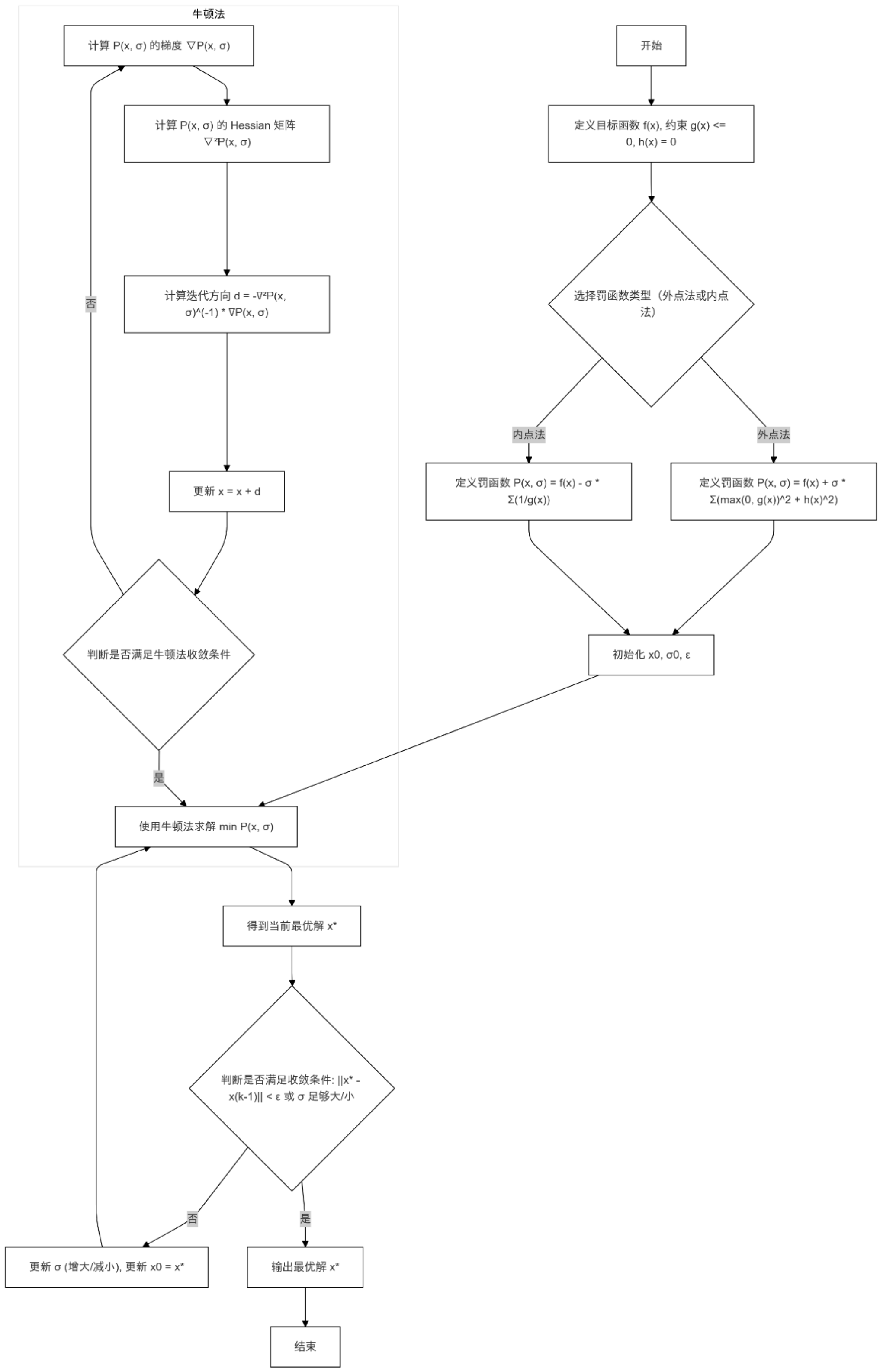
班级：信息学部 8 班

哈尔滨工业大学（深圳）

2024.12

作业一

一、 设计程序框图



牛顿法
计算 P(x, σ) 的梯度 ∇P(x, σ)
计算 P(x, σ) 的 Hessian 矩阵 ∇²P(x, σ)
计算迭代方向 d = -∇²P(x, σ)^(-1) * ∇P(x, σ)
更新 x = x + d
判断是否满足牛顿法收敛条件
否
是
使用牛顿法求解 min P(x, σ)
开始
定义目标函数 f(x), 约束 g(x) <= 0, h(x) = 0
选择罚函数类型（外点法或内点法）
内点法
外点法
定义罚函数 P(x, σ) = f(x) - σ * Σ(1/g(x))
定义罚函数 P(x, σ) = f(x) + σ * Σ(max(0, g(x))^2 + h(x)^2)
初始化 x0, σ0, ε
得到当前最优解 x*
判断是否满足收敛条件: ||x* - x(k-1)|| < ε 或 σ 足够大/小
否
是
更新 σ (增大/减小), 更新 x0 = x*
输出最优解 x*
结束

二、 求解程序

使用 Python 语言编写，程序如下：

```
import numpy as np
from scipy.optimize import minimize


def constrained_optimization(f, x0, g=[], h=[],
sigma0=1.0, epsilon=1e-6, max_iter=100):
    """
    使用外点罚函数法和牛顿法求解约束优化问题

    Args:
        f: 目标函数
        x0: 初始点
        g: 不等式约束函数列表，每个函数表示一个 g(x) <= 0 的约
束
        h: 等式约束函数列表，每个函数表示一个 h(x) = 0 的约束
        sigma0: 初始罚因子
        epsilon: 收敛精度
        max_iter: 最大迭代次数

    Returns:
        x_star: 最优解
    """

    sigma = sigma0
    x_prev = x0

    def penalty_function(x, sigma):
        """外点罚函数"""
        penalty = 0
        for gi in g:
            penalty += max(0, gi(x)) ** 2
        for hi in h:
            penalty += hi(x) ** 2
        return f(x) + sigma * penalty

    for i in range(max_iter):
        # 使用牛顿法求解无约束优化问题
        result = minimize(
            lambda x: penalty_function(x, sigma),
            x_prev,
            method="Newton-CG",
            jac=lambda x: approx_gradient(lambda x:
penalty_function(x, sigma), x),
```

```
            hess=lambda x: approx_hessian(lambda x:
penalty_function(x, sigma), x),
        )
        x_star = result.x

        # 判断收敛条件
        if np.linalg.norm(x_star - x_prev) < epsilon or
sigma > 1e10:
            print(f"迭代次数: {i+1}")
            return x_star

        # 更新罚因子和初始点
        sigma *= 10
        x_prev = x_star

    print(f"达到最大迭代次数 {max_iter}")
    return x_star


def approx_gradient(f, x, h=1e-5):
    """数值方法计算梯度"""
    grad = np.zeros_like(x)
    for i in range(len(x)):
        x_plus = x.copy()
        x_plus[i] += h
        x_minus = x.copy()
        x_minus[i] -= h
        grad[i] = (f(x_plus) - f(x_minus)) / (2 * h)
    return grad


def approx_hessian(f, x, h=1e-5):
    """数值方法计算 Hessian 矩阵"""
    n = len(x)
    hess = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            x_pp = x.copy()
            x_pp[i] += h
            x_pp[j] += h

            x_pm = x.copy()
            x_pm[i] += h
            x_pm[j] -= h

            x_mp = x.copy()
```

```
            x_mp[i] -= h
            x_mp[j] += h

            x_mm = x.copy()
            x_mm[i] -= h
            x_mm[j] -= h

            hess[i, j] = (f(x_pp) - f(x_pm) - f(x_mp) +
f(x_mm)) / (4 * h**2)
    return hess


# 程序输入、输出说明:
# 输入:
#   - f: 目标函数，接受一个 numpy 数组作为输入，返回一个标量
#   - x0: 初始点，一个 numpy 数组
#   - g: 不等式约束函数列表，每个函数接受一个 numpy 数组作为输
入，返回一个标量，默认为空列表
#   - h: 等式约束函数列表，每个函数接受一个 numpy 数组作为输入，
返回一个标量，默认为空列表
#   - sigma0: 初始罚因子，一个标量，默认为 1.0
#   - epsilon: 收敛精度，一个标量，默认为 1e-6
#   - max_iter: 最大迭代次数，一个整数，默认为 100
# 输出:
#   - x_star: 最优解，一个 numpy 数组
```

三、 算例描述

为方便检验正确性，采用作业中的一道题作为算例：

$$\begin{cases} min\ \ f(\mathbf{x}) = (x_1 - \dfrac{9}{4})^2 + (x_2 - 2)^2 \\ s.t.\ \ x_2 - x_1^2 \geq 0 \\ \qquad\ \ x_1 + x_2 \leq 6 \\ \qquad\ \ x_1, x_2 \geq 0 \end{cases}$$

```
# 定义目标函数
def f(x):
    return (x[0] - 9.0 / 4) ** 2 + (x[1] - 2) ** 2


# 定义不等式约束
def g1(x):
    return x[0] ** 2 - x[1]


def g2(x):
```

```python
    return x[0] + x[1] - 6


def g3(x):
    return -x[0]


def g4(x):
    return -x[1]


# 定义等式约束（无）

# 初始点
x0 = np.array([0.0, 0.0])

# 调用函数求解
x_star = constrained_optimization(f, x0, [g1, g2, g3,
g4])

print("最优解:", x_star)
print("目标函数值:", f(x_star))
print("g1(x*):", g1(x_star))
print("g2(x*):", g2(x_star))
print("g3(x*):", g3(x_star))
print("g4(x*):", g4(x_star))
```

四、 算例验证及论证说明

运行程序得到输出结果如下：

迭代次数: 7

最优解: [1.4999638 2.24989323]

目标函数值: 0.6250009332834441

g1(x*): -1.8398370307259881e-06

g2(x*): -2.2501429685492647

g3(x*): -1.4999637975757723

g4(x*): -2.249893233874963

作业中使用的数学方法如下：

1. 问题

$$\begin{cases}\min f(\boldsymbol{x})=\left(x_1-\frac{9}{4}\right)^2+(x_2-2)^2\\ s.t.\ x_2-x_1^2\ge 0\\ \qquad x_1+x_2\le 6\\ \qquad x_1,x_2\ge 0\end{cases}$$

验证 $\boldsymbol{x}^*=(1.5,2.25)^T$ 是 K-T 点。

解：

$$\begin{cases}g_1(\boldsymbol{x})=x_1^2-x_2\\ g_2(\boldsymbol{x})=x_1+x_2-6\\ g_3(\boldsymbol{x})=-x_1\\ g_4(\boldsymbol{x})=-x_2\end{cases}$$

显然仅有 $g_1(\boldsymbol{x}^*)=0$，$I=\{1\}$，计算梯度

$$\nabla f(\boldsymbol{x})=\left[2x_1-\frac{9}{2}\ \ 2x_2-4\right]^T$$

$$\nabla g_1(\boldsymbol{x})=[2x_1\ \ -1]^T$$

将 $\boldsymbol{x}^*$ 代入得

$$\begin{cases}-\frac{3}{2}+3u_1=0\\ \frac{1}{2}-u_1=0\end{cases}$$

解得 $u_1=\frac{1}{2}\ge 0$，故 $\boldsymbol{x}^*=(1.5,2.25)^T$ 是 K-T 点。

观察到，迭代收敛之后所得到的解与数学方法得到的解析解［1.5, 2.25］近似相同，可以认为程序正确完成了任务。

作业二

一、 问题描述

大学生阶段是个人成长和发展的关键时期，面临着学业、社团活动、个人兴趣、社交、兼职、休息等多种任务和需求的竞争。如何合理分配有限的时间资源，以最大化个人效用，是每个大学生都需要面对的挑战。

具体来说，一个典型的大学生每天需要考虑以下几个方面的时间分配：

- 学习时间：包括上课、自习、完成作业、阅读文献等。学习时间直接影响学业成绩，是大学生最重要的任务之一。
- 社团活动时间：参加学生组织、社团活动可以提升个人能力、拓展人脉，但也会占用一定的时间。
- 个人兴趣时间：发展个人兴趣爱好，如运动、音乐、阅读等，可以放松身心，提升生活质量。
- 社交时间：与朋友、家人、同学的交流互动，是维持人际关系、获取情感支持的重要途径。
- 休息娱乐时间：充足的睡眠和休息是保证身心健康、提高学习效率的基础。

这些不同的活动和任务之间存在着复杂的相互影响关系。例如，过多的学习时间可能会挤占休息时间，导致疲劳和效率下降；而过多的社交时间可能会影响学习成绩。此外，每个人的精力、能力和目标都不同，因此最优的时间分配方案也因人而异。因此，如何根据自身的实际情况，制定一个合理的时间分配方案，以最大化个人在学业、能力、身心健康等方面的综合收益，是一个值得深入研究的优化问题。

二、 数学模型建立与描述

为了简化问题，我们假设一个大学生每天需要分配时间到以下五种活动：学习(x1)、社团活动(x2)、个人兴趣(x3)、社交(x4)和休息娱乐(x5)。

目标函数：

我们的目标是最大化个人效用 U，可以将其定义为各个活动带来的效用之和。假设每个活动的效用与其投入时间成正比，但由于边际效用递减，我们可以使用对数函数来表示这种关系：

U(x1,x2,x3,x4,x5)=w1*ln(x1+1)+w2*ln(x2+1)+w3*ln(x3+1)+w4*ln(x4+1)+w5*ln(x5+1)

其中，w1,w2,w3,w4,w5 分别代表各个活动的权重，反映了它们对个人效用的相

对重要程度。这些权重可以根据个人的价值观和目标进行设定。例如，一个重视学业的学生可能会将 w1 设置得较高。加 1 是为了避免对数函数在 0 处无定义。

约束条件：

时间约束：每天的总时间是有限的，x1+x2+x3+x4+x5=24

非负约束：每个活动的时间分配不能为负，x1>=0, x2>=0, x3>=0, x4>=0, x5>=0

最低休息时间约束：为了保证身心健康，每天需要保证一定的最低休息时间，假设为 6 小时，x5>=6

综上，该优化问题可描述为：

$$max\, U = w_1\, ln(x_1+1) + w_2\, ln(x_2+1) + w_3\, ln(x_3+1) + w_4\, ln(x_4+1) + w_5\, ln(x_5+1)$$

$$s.t. \begin{cases} x_1+x_2+x_3+x_4+x_5 = 24 \\ x_5 \geq 6 \\ x_1, x_2, x_3, x_4, x_5 \geq 0 \end{cases}$$

三、 求解方法与求解过程

这是一个带约束的非线性规划问题，我们可以使用拉格朗日乘子法结合数值优化方法进行求解。

构建拉格朗日函数：

L(x1, x2, x3, x4, x5, λ, μ)=w1*ln(x1+1)+w2*ln(x2+1)+w3*ln(x3+1)+w4*ln(x4+1)+w5*ln(x5+1)-λ(x1+x2+x3+x4+x5-24)-μ(6-x5)

分别求解偏导数并令其等于 0，注意 μ=0（x5>6 时）。由于方程组是非线性的，难以直接求得解析解，我们可以使用数值优化方法，例如梯度下降法或牛顿法，结合编程工具（如 Python 的 SciPy 库）进行迭代求解。

设定参数：

假设一个重视学业和个人发展的学生，可以将权重设置为 w1=0.4, w2=0.1, w3=0.2, w4=0.1, w5=0.2

使用 Python 进行求解：

```python
import numpy as np
from scipy.optimize import minimize


# 定义目标函数
def objective(x, w):
    return -(
        w[0] * np.log(x[0] + 1)
        + w[1] * np.log(x[1] + 1)
        + w[2] * np.log(x[2] + 1)
        + w[3] * np.log(x[3] + 1)
```

```python
            + w[4] * np.log(x[4] + 1)
    )


# 定义约束条件
def constraint1(x):
    return x[0] + x[1] + x[2] + x[3] + x[4] - 24


def constraint2(x):
    return x[4] - 6


# 设定权重
w = [0.4, 0.1, 0.2, 0.1, 0.2]

# 设定初始值
x0 = [4.8, 4.8, 4.8, 4.8, 4.8]

# 定义约束条件类型和边界
con1 = {"type": "eq", "fun": constraint1}
con2 = {"type": "ineq", "fun": constraint2}
cons = [con1, con2]
bnds = [(0, 24), (0, 24), (0, 24), (0, 24), (0, 24)]

# 使用 minimize 函数求解
solution = minimize(
    objective, x0, args=(w,), method="SLSQP",
bounds=bnds, constraints=cons
)

# 打印结果
print(solution)
```

四、 结果讨论与分析

运行程序，获得输出结果如下：

```
message: Optimization terminated successfully
 success: True
  status: 0
     fun: -1.8915986277783705
       x: [ 9.973e+00  1.738e+00  4.550e+00  1.738e+00  6.000e+00]
     nit: 7
     jac: [-3.645e-02 -3.652e-02 -3.603e-02 -3.652e-02 -2.857e-02]
```

```
nfev: 42
njev: 7
```

结果分析：

- 学习时间占比最高：在给定的权重下，学习时间占据了最大比例，约为 9.973 小时，这与我们设定的学生重视学业的假设相符。
- 休息时间得到保障：优化结果显示休息时间为 6 小时，满足设定的约束条件。
- 各项活动均有涉及：尽管学习时间占比最高，但其他活动也都有一定的时间分配，这表明一个平衡的生活方式有助于提升整体效用。
- 权重的影响：不同的权重设置会导致不同的最优解。例如，本例中社团活动和社交的权重值相同，得到的分配时间也相同。这说明该模型可以根据个人的实际情况进行调整，从而得到个性化的时间管理方案。

模型局限性：

- 效用函数的简化：我们使用对数函数来表示效用，这是一种简化的假设。实际上，不同活动的效用函数可能更加复杂，而且不同活动之间可能存在相互影响。
- 权重的确定：权重的设定具有一定的主观性，如何准确地量化不同活动的相对重要程度是一个挑战。
- 动态变化：大学生的时间管理需求可能会随着学期、考试周等因素发生变化，而该模型是一个静态模型，没有考虑这些动态变化。
- 个体差异：模型的求解结果是基于平均情况的，没有考虑个体在精力、学习效率等方面的差异。

模型改进方向：

- 引入更复杂的效用函数：可以考虑使用其他类型的函数，如二次函数，或者引入交叉项来表示不同活动之间的相互影响。
- 使用问卷调查等方法确定权重：可以通过问卷调查等方式，收集更多学生的数据，从而更准确地估计权重参数。
- 建立动态模型：可以将时间划分为更小的单位，例如周或月，并考虑不同时间段的任务和需求变化，建立动态优化模型。
- 考虑个体差异：可以引入更多变量来描述学生的个体特征，例如学习效率、精力水平等，从而得到更精细化的时间管理方案。

总结：

本案例展示了如何将一个实际生活中的问题转化为数学优化问题，并使用最优化方法进行求解。尽管模型存在一定的局限性，但它为大学生时间管理提供了一个有益的参考框架。通过调整模型参数和约束条件，可以得到个性化的时间分配方案，从而帮助大学生更有效地利用时间，实现个人目标。同时，该案例也说

明了最优化方法在解决实际问题中的应用价值，以及不断改进模型以提高其准确性和实用性的重要性。