

Lab1 实验报告

目标函数与约束条件

目标函数：

$$\text{Minimize} \quad \sum_{i,j} (x_{i,j} \times \text{image}[i,j])^2 + \sum_{i,j} ((1 - x_{i,j}) \times (1 - \text{image}[i,j]))^2$$

其中，image[i,j] 中存的是已知的数组，为 0 到 1 的小数（定为 0.01 与 0.99）。x[i,j] 是存的需要求解的变量，在本次事项中使用布尔变量作为保存的 variable。

约束条件：

x[i,j] 的取值只能为 0 或 1。（因此在实现中使用一个布尔变量作为其取值）

代码的实现逻辑

首先定义了一个函数 solveProblem 作为问题的求解器，以 image 函数和当前输入图像的类型作为输入。然后先将初始的图片输出，然后设置目标函数和一些要用到的常数，通过求解器进行求解。由于求解得到的 x 数组与真实需要打印的结果之和为 1，即得到的 x 数组表示的是分离出的背景，因此要输出分离出的图案需要再进行一次减操作。

```
def solveProblem(image, name):
    # 保存原始图像到文件
    plt.title(name + " Original Picture")
    plt.imshow(image, cmap='gray')
    plt.savefig(name + "Original Picture.png")

    # 设置变量、常量、优化目标，并求解
    ones = np.ones((size, size))
    onesConstant = cp.Constant(ones)
    imgConstant = cp.Constant(image)
    x = cp.Variable((size, size), boolean = True)
    objective = cp.Minimize(cp.sum(cp.multiply(x, cp.square(imgConstant))) +
                             cp.sum(cp.multiply(onesConstant - x, cp.square(onesConstant - imgConstant))))
    problem = cp.Problem(objective)
    problem.solve(solver = cp.ECOS_BB)

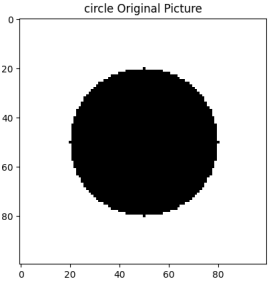

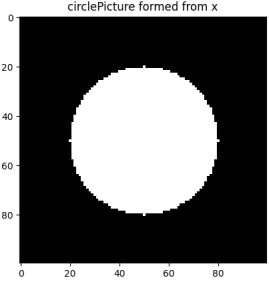

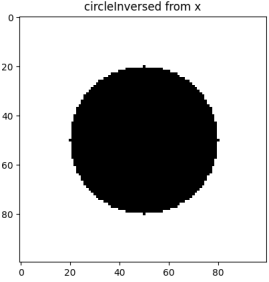

    # 保存结果图像到文件
    plt.title(name + "Picture formed from x")
    plt.imshow(x.value, cmap='gray')
    plt.savefig(name + "formed Picture.png")

    x = ones - x
    plt.title(name + "Inversed from x")
    plt.imshow(x.value, cmap='gray')
    plt.savefig(name + "Analized Picture.png")
```

此外，也定义了两个生成图片的 create_image 函数，通过这两个函数分别生成了要被拟合的圆形和三角形图案。而在真正的运行过程中，则是先调用图案生成器，再调用 solver 进行求解并且输出要求的信息。

结果展示

实验结果如下表所示：

图形类型	圆形	三角形
原始图像		
用直接解得的 x数组生成的 图像		
用1- x 生成的 图像 (segmented Image)		

从以上图像可以看出，在使用 solver 计算出来的 x 确实将图案与背景分离了出来。同时，在我们取的值为 0.99 与 0.01 时，目标函数的两个部分的贡献是比较均衡的，分离出来的图像也非常清晰，与原图完全一致。但是，当图像中的原始数据取值开始接近时，分离出的图像就会发生变化，而且会出现最终分离出的图片错误的情况。

举个例子，当使用如下函数作为生成原始图像的函数，其分离出的图像就会出现异常。这说明在不同情况下，目标函数的选取需要做相应的改变，每部分占到的权重要有所区分。

```
def create_image_triangle(size, base):
    image = np.ones((size, size)) * 0.99
    for i in range(size):
        for j in range(size):
            if i > size / 2 - base / 4 and i < size / 2 + base / 4 and \
                j > size / 2 - i + base / 2 and j < size / 2 + i - base / 2:
                image[i,j] = 0.01 * (i / 2 + j / 3)
    return image
```