Assignment #2: 深度学习与大语言模型

Updated 2204 GMT+8 Feb 25, 2025

2025 spring, Complied by <mark>汤伟杰，信息管理系</mark>

作业的各项评分细则及对应的得分

标准	等级	得分
按时提交	完全按时提交：1 分 提交有请假说明：0.5 分 未提交：0 分	1 分
源码、耗时（可选）、解题思路（可选）	提交了 4 个或更多题目且包含所有必要信息：1 分 提交了 2 个或以上题目但不足 4 个：0.5 分 少于 2 个：0 分	1 分
AC 代码截图	提交了 4 个或更多题目且包含所有必要信息：1 分 提交了 2 个或以上题目但不足 4 个：0.5 分 少于：0 分	1 分
清晰头像、PDF 文件、MD/DOC 附件	包含清晰的 Canvas 头像、PDF 文件以及 MD 或 DOC 格式的附件：1 分 缺少上述三项中的任意一项：0.5 分 缺失两项或以上：0 分	1 分
学习总结和个人收获	提交了学习总结和个人收获：1 分 未提交学习总结或内容不详：0 分	1 分
总得分： 5	总分满分：5 分	

说明：

1. **解题与记录：**

    - 对于每一个题目，请提供其解题思路（可选），并附上使用 Python 或 C++ 编写的源代码（确保已在 OpenJudge， Codeforces，LeetCode 等平台上获得 Accepted）。请将这些信息连同显示“Accepted”的截图一起填写到下方的作业模板中。（推荐使用 Typora https://typoraio.cn 进行编辑，当然你也可以选择 Word。）无论题目是否已通过，请标明每个题目大致花费的时间。

2. **课程平台与提交安排：**

- 我们的课程网站位于 Canvas 平台（https://pku.instructure.com ）。该平台将在第 2 周选课结束后正式启用。在平台启用前，请先完成作业并将作业妥善保存。待 Canvas 平台激活后，再上传你的作业。

    - 提交时，请首先上传 PDF 格式的文件，并将.md 或.doc 格式的文件作为附件上传至右侧的“作业评论”区。确保你的 Canvas 账户有一个清晰可见的头像，提交的文件为 PDF 格式，并且“作业评论”区包含上传的.md 或.doc 附件。

3. **延迟提交：**

- 如果你预计无法在截止日期前提交作业，请提前告知具体原因。这有助于我们了解情况并可能为你提供适当的延期或其他帮助。

请按照上述指导认真准备和提交作业，以保证顺利完成课程要求。

1. 题目

18161: 矩阵运算

matrices, http://cs101.openjudge.cn/practice/18161

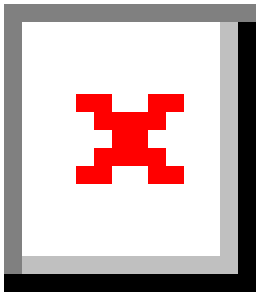
思路：

两层循环遍历矩阵即可，对 c 矩阵原地操作。

代码：

```
an,am=map(int,input().split())
a=[]
for _ in range(an): a.append(list(map(int,input().split())))
bn,bm=map(int,input().split())
b=[]
for _ in range(bn): b.append(list(map(int,input().split())))
cn,cm=map(int,input().split())
c=[]
for _ in range(cn): c.append(list(map(int,input().split())))
if am!=bn or an!=cn or bm!=cm:
    print('Error!')
else:
    for i in range(an):
        for j in range(bm):
            c[i][j]+=sum(a[i][k]*b[k][j] for k in range(am))
    for i in c:
        print(*i)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



19942: 二维矩阵上的卷积运算

matrices, http://cs101.openjudge.cn/practice/19942/

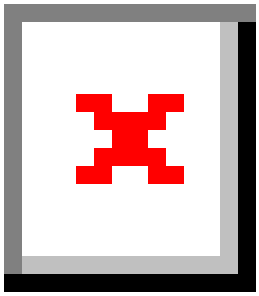
思路：

四层循环即可，还以为会超时，没想到数据这么小。

代码：

```
m,n,p,q=map(int,input().split())
mn=[list(map(int,input().split())) for _ in range(m)]
pq=[list(map(int,input().split())) for _ in range(p)]
ans=[[0]*(n+1-q) for _ in range(m+1-p)]
for x in range(m+1-p):
    for y in range(n+1-q):
        for i in range(p):
            for j in range(q):
                ans[x][y]+=pq[i][j]*mn[x+i][y+j]
for i in ans:
    print(' '.join(map(str,i)))
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



04140: 方程求解

牛顿迭代法，http://cs101.openjudge.cn/practice/04140/

请用<mark>牛顿迭代法</mark>实现。

因为大语言模型的训练过程中涉及到了梯度下降（或其变种，如 SGD、Adam 等），用于优化模型参数以最小化损失函数。两种方法都是通过迭代的方式逐步接近最优解。每一次迭代都基于当前点的局部信息调整参数，试图找到一个比当前点更优的新点。理解牛顿迭代法有助于深入理解基于梯度的优化算法的工作原理，特别是它们如何利用导数信息进行决策。

牛顿迭代法

- **目的**：主要用于寻找一个函数 $f(x)$ 的根，即找到满足 $f(x)=0$ 的 x 值。不过，通过适当变换目标函数，它也可以用于寻找函数的极值。
- **方法基础**：利用泰勒级数的一阶和二阶项来近似目标函数，在每次迭代中使用目标函数及其导数的信息来计算下一步的方向和步长。
- **迭代公式**：$ x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} $ 对于求极值问题，这可以转化为 $ x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)} $，这里 $f'(x)$ 和 $f''(x)$ 分别是目标函数的一阶导数和二阶导数。
- **特点**：牛顿法通常具有更快的收敛速度（尤其是对于二次可微函数），但是需要计算目标函数的二阶导数（Hessian 矩阵在多维情况下），并且对初始点的选择较为敏感。

梯度下降法

- **目的**：直接用于寻找函数的最小值（也可以通过取负寻找最大值），尤其在机器学习领域应用广泛。

- **方法基础**：仅依赖于目标函数的一阶导数信息（即梯度），沿着梯度的反方向移动以达到减少函数值的目的。
- **迭代公式**：$ x_{n+1} = x_n - \alpha \cdot \nabla f(x_n) $ 这里 α 是学习率，$\nabla f(x_n)$ 表示目标函数在 x_n 点的梯度。
- **特点**：梯度下降不需要计算复杂的二阶导数，因此在高维空间中相对容易实现。然而，它的收敛速度通常较慢，特别是当目标函数的等高线呈现出椭圆而非圆形时（即存在条件数大的情况）。

相同与不同

- **相同点**：两者都可用于优化问题，试图找到函数的极小值点；都需要目标函数至少一阶可导。
- **不同点**：
  - 牛顿法使用了更多的局部信息（即二阶导数），因此理论上收敛速度更快，但在实际应用中可能会遇到计算成本高、难以处理大规模数据集等问题。
  - 梯度下降则更为简单，易于实现，特别是在高维空间中，但由于只使用了一阶导数信息，其收敛速度可能较慢，尤其是在接近极值点时。

思路：

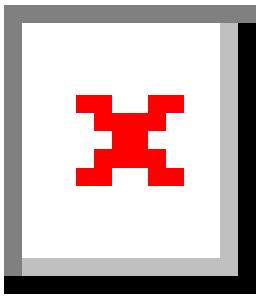
参考了 gpt 的代码，把 fx 和 dfx 作为参数传入主函数 newton_method 中，方便维护。同时设置了最大迭代次数来防止无法收敛的情况。

代码：

```
def f(x):
    return x**3-5*x**2+10*x-80
def df(x):
    return 3*x**2-10*x+10

def newton_method(f,df,x0,tol=1e-9,max_iter=10000):
    x=x0
    for i in range(max_iter):
        fx=f(x)
        dfx=df(x)
        if dfx==0:
            return None
        x_new=x-fx/dfx
        if abs(x_new-x)<tol:
            return x_new

        x=x_new

    return None

root=newton_method(f,df,1.0)
if root is not None:
    print(f'{root:.9f}')
```

```
else:
    print('Error!')
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



06640: 倒排索引

data structures, http://cs101.openjudge.cn/practice/06640/

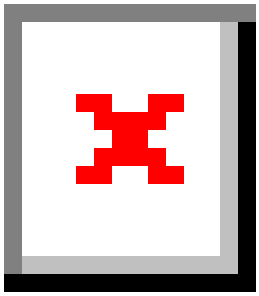
思路：

使用 defaultdict(set)来存每个单词存在的文档编号，坑点在于可能一句话里面出现了多次相同的单词，如果不用 set 可能会忘记删去重复的编号，用 list 会使效率降低。

代码：

```
from collections import defaultdict
n=int(input())
d=defaultdict(set)

for idx in range(1,n+1):
    s=list(input().split())
    for word in s[1:]:
        if idx not in d[word]:
            d[word].add(idx)

m=int(input())
for _ in range(m):
    w=input()
    if w in d:
        print(' '.join(map(str,sorted(d[w]))))
    else:
        print('NOT FOUND')
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



04093: 倒排索引查询

data structures, http://cs101.openjudge.cn/practice/04093/

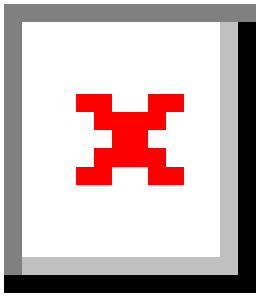
思路：

结合上面一题，这道题就是集合的交集和差集运算。先找到一个 1 作为基准，接下来遇到 0 跳过，遇到 1 求交集，遇到-1 求差集即可。

代码：

```
n=int(input())
d={}
for _ in range(n):
    s=list(map(int,input().split()))
    d[_]=set(s[1:])

m=int(input())
for _ in range(m):
    l=list(map(int,input().split()))
    idx=l.index(1)
    start=d[idx].copy()
    for num in range(n):
        if l[num]==1:
            start&=d[num]
        elif l[num]==-1:
            start-=d[num]
    if start:
        print(*sorted(start))
    else:
        print('NOT FOUND')

#
# a,b={1,2,3},{1,2}
# print(a&b)
# print(a-b)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



Q6. Neural Network 实现鸢尾花卉数据分类

在 http://clab.pku.edu.cn 云端虚拟机，用 Neural Network 实现鸢尾花卉数据分类。

参考链接，https://github.com/GMyhf/2025spring-cs201/blob/main/LLM/iris_neural_network.md

2. 学习总结和个人收获

<mark>如果发现作业题目相对简单，有否寻找额外的练习题目，如“数算 2025spring 每日选做”、LeetCode、Codeforces、洛谷等网站上的题目。</mark>

倒排索引是我上学期专业课的内容，能在题目里面遇到相当惊讶，我老师说顺排和倒排索引对于大一学生来说比较难理解，幸好我提前做了功课才能比较顺利地解决这道题。不过这两道题好像都是在考察集合的知识？

这学期选了程序设计课，同时在学习 c 的 oop 知识，感觉 python 写代码是真心简单，c 语言太复杂太麻烦了。。。用 python 写 100 行代码我感觉已经很长了，结果学 c 的同学说 100 行代码是很正常的？？？

每日选做还在跟进！