Assignment #2: 深度学习与大语言模型

Updated 2204 GMT+8 Feb 25, 2025

2025 spring, Complied by 叶靖、信管

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：

其实这题的难点就是在矩阵的乘法，要将行和列分别进行乘法。 另外，刚巧在其他课程中上到numpy，所以自己尝试了用numpy的功能去解题（如下），认真方便很多。

import numpy as np

row_A, column_A = map(int, input().split()) A = [list(map(int, input().split())) for _ in range(row_A)]

row_B, column_B = map(int, input().split()) B = [list(map(int, input().split())) for _ in range(row_B)]

row_C, column_C = map(int, input().split()) C = [list(map(int, input().split())) for _ in range(row_C)]

if column_A != row_B or row_C != row_A or column_B != column_C: print('Error!')

else: A = np.array(A) B = np.array(B) C = np.array(C)

```
ans = np.dot(A, B) + C

for row in ans:
    print(' '.join(map(str, row)))
```

代码：

def matrix_sum(A, B): row_A, col_A = len(A), len(A[0]) ans_1 = [[0] * col_A for _ in range(row_A)]

```
for i in range(row_A):
    for j in range(col_A):
```

```
            ans_1[i][j] = A[i][j] + B[i][j]
    return ans_1
```

def matrix_multiply(A, B): row_A, col_A = len(A), len(A[0]) row_B, col_B = len(B), len(B[0]) ans_2 = [[0] * col_B for _ in range(row_A)]

```
    for i in range(row_A):
        for j in range(col_B):
            for k in range(col_A):
                ans_2[i][j] += A[i][k] * B[k][j]

    return ans_2
```

row_A, column_A = map(int, input().split()) A = [list(map(int, input().split())) for _ in range(row_A)]

row_B, column_B = map(int, input().split()) B = [list(map(int, input().split())) for _ in range(row_B)]

row_C, column_C = map(int, input().split()) C = [list(map(int, input().split())) for _ in range(row_C)]

if column_A != row_B or row_C != row_A or column_B != column_C: print('Error!')

else: ans = matrix_multiply(A, B) result = matrix_sum(ans, C) for row in result: print(' '.join(map(str, row)))

代码运行截图 （至少包含有"Accepted"）

 02-01

19942: 二维矩阵上的卷积运算

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

思路：

代码：

代码运行截图 （至少包含有"Accepted"）

04140: 方程求解

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

请用牛顿迭代法实现。

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：

看了老师给的补充资料，但还是不大理解，所以有寻求大模型的帮忙，但我觉得应该还能优化目前的代码。

代码：

```
import random

def func(x): return x ** 3 - 5 * x ** 2 + 10 * x - 80

def f_prime(x): # a方程式导数（d/dx） return 3 * x ** 2 - 10 * x + 10
```

def newton(initial, tolerance=1e-6, max_iterations=100): x_n = initial for n in range(max_iterations): fx_n = func(x_n) f_prime_x_n = f_prime(x_n)

```
    if f_prime_x_n == 0:
        return None

    x_n1 = x_n - fx_n / f_prime_x_n
    if abs(x_n1 - x_n) < tolerance:
        return x_n1

    x_n = x_n1

return x_n
```

initial = random.random() ans = newton(initial) print(f'')

代码运行截图 （至少包含有"Accepted"）

 02-02

06640: 倒排索引

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

思路：

代码：

row_num = int(input()) words = [] for i in range(row_num): inserts = input().split() words.append((i + 1, inserts))

question_num = int(input()) for _ in range(question_num): finding_word = input() found = False ans = [] for row_num, word_list in words: if finding_word in word_list: ans.append(str(row_num)) found = True if not found: ans.append('NOT FOUND')

```
print(' '.join(ans))
```

代码运行截图 （至少包含有"Accepted"）

 02-03

04093: 倒排索引查询

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

思路：

代码：

def document_finding(numbers, command, n): storage = None delete_num = set() for i in range(n): if command[i] == 1: if storage is None: storage = set(numbers[i]) else: storage &= set(numbers[i]) elif command[i] == -1: delete_num |= set(numbers[i]) elif command[i] == 0: pass if storage is None: return 'NOT FOUND'

```
ans = sorted(storage - delete_num)
return ans if ans else "NOT FOUND"
```

n = int(input()) numbers = [] for _ in range(n): inserts = input().split() numbers.append(set(map(int, inserts[1:])))

m = int(input()) for _ in range(m): commands = list(map(int, input().split())) ans = document_finding(numbers, commands, n) if ans == 'NOT FOUND': print(ans) else: print(' '.join(map(str, ans)))

代码运行截图 （至少包含有"Accepted"）

 02-04

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

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。

这次的作业做起来明显比上次的作业难度增加了不少，但能明显感觉到这次作业专注于矩阵和倒排索引。 尤其是在倒排索引的题目上能感受到自己做的过程更为吃力，所以后续可能会集中刷这类型的题目。有时在做矩阵的时候会不自觉地想和其他课程内容联想，感觉蛮奇妙的。