最优化理论与方法期末课程实验报告

实验者： 何峙
学号： 21215122
专业： 大数据与人工智能

实验目标

设计一种算法计算矩阵所有特征值与特征向量。

实验步骤

设有矩阵 A 为 n 阶方阵（$A \in \mathbb{R}^{n\times n}$），现要求 A 的所有特征值与特征向量。

1. 使用如下幂法迭代式求矩阵 A 的最大特征值 eigval 及其对应特征向量 eigvec：

$$u_k = \frac{v_k}{\|v_k\|_\infty}, v_{k+1} = Au_k$$

$$\begin{cases} eigvec = \lim\limits_{k\to+\infty} u_k \\ eigval = \lim\limits_{k\to+\infty} \|v_k\|_\infty \end{cases}$$

代码如下：

```
def powerIteration(m):
    x = torch.rand((m.size()[0],))
    n = 0
    err = float('inf')
    x_m = torch.max(x)
    eig_vec = torch.zeros((m.size()[0],))
    eig_val = 0
    while n < N_MAX and err >= VAL_ERR:
        x_u = x / x_m
        x = torch.matmul(m, x_u)
        eig_val = torch.max(x)
        eig_vec = x_u
        err = abs(eig_val - x_m)
        x_m = eig_val
        n += 1
    return eig_vec, eig_val
```

2. 使用特征向量计算对应的 Householder 矩阵 H，并对矩阵 A 做正交相似变换：

$$B = HAH^T$$

3. 取以上变换后的矩阵 B 右下角子矩阵(去掉第一行与第一列后剩下的矩阵)作为新的矩阵 A，重复步骤 1、2，直到矩阵 A 只剩一个元素，则结束，代码如下：

```
def main(mat):
    n = mat.size()[0]
    eig_vec_list = []
    eig_val_list = []
    H_list = []
    r1_list = []
    evectmp_list = []
    for i in range(n):
        eig_vec, eig_val = powerIteration(mat)
        eig_val_list.append(eig_val)

        if i == 0:
            eig_vec_list.append(eig_vec)
        else:
            evectmp_list.append(eig_vec)
            eval_idx = len(eig_val_list) - 1
            for H_mat, r1_vec, evectmp in zip(reversed(H_list), reversed(r1_list),
reversed(evectmp_list)):
                alpha = r1_vec.dot(evectmp) / (eig_val_list[eval_idx] -
eig_val_list[eval_idx - 1])
                alpha = alpha.unsqueeze(0)
                tmp_vec = torch.cat((alpha, evectmp)).view(-1, 1)
                final_vec = torch.matmul(H_mat, tmp_vec)

        H = getHouseholderMatrix(eig_vec) #  获得 Householder 矩阵
        A = torch.matmul(torch.matmul(H, mat), H.mT)   #  用 Householder 矩阵对原始矩
阵做正交相似变换

        mat = A[1:, 1:]
        r1 = A[0, 1:]
        H_list.append(H)
        r1_list.append(r1)
        if mat.size()[0] == 1:
            eig_val_list.append(mat[0][0])
            break

    return eig_val_list, eig_vec_list
```

实验结果

以一下矩阵为例，计算其所有特征值：

$$A = \begin{bmatrix} 3, -1, 1 \\ 2\ , 0\ , 1 \\ 1, -1, 2 \end{bmatrix}$$

计算所有特征值结果为：

[2.0364983   1.9635155   0.99999976]

而调用 python 接口 numpy.linalg.eig 做验证对比，其结果为：

[2. 2. 1.]

可见在误差范围内两者几乎一样，算法是通过的。
在计算出所有特征值后，可用消元法解线性方程组，得出所有特征向量，这里不做另外展示了。

实验分析

本实验使用收缩技术，假设已经求出了矩阵 A 的一个特征值λ_1及相应特征向量 x1, 即：

$$Ax_1 = \lambda_1 x_1$$

然后构造一个 Householder 变换矩阵，使得：

$$Hx_1 = \sigma e_1, \quad e_1 = [1, 0, ..., 0]^T$$

然后可得：

$$HAH^T e_1 = HA(\frac{1}{\sigma})x_1 = \frac{1}{\sigma} HAx_1 = \frac{1}{\sigma} H\lambda_1 x_1 = \frac{\lambda_1}{\sigma}(\sigma e_1) = \lambda_1 e_1$$

易知上式中 $HAH^T e_1$即为HAH^T的第一列，所以有：

$$HAH^T = \begin{bmatrix} \lambda_1, r_1^T \\ \mathbf{0}\ , A_1 \end{bmatrix}$$

其中 $A_1 \in \mathbb{R}^{(n-1)\times(n-1)}$, $r_1 \in \mathbb{R}^{n-1}$。

因为 Householder 矩阵 H 是正交矩阵，而正交相似变换不改变矩阵的特征值，所以求矩阵 A 的其余特征值变为求 n-1 阶矩阵 A_1的特征值，那么如此不断的对所求矩阵进行收缩，对收缩后的矩阵继续用幂法，即可求出所有特征值。

附：实验编程代码

```
import numpy as np
import torch

N_MAX = 50
VAL_ERR = 1e-5

# 用幂法求矩阵 m 的最大特征值及对应特征向量
def powerIteration(m):
    x = torch.rand((m.size()[0],))
    n = 0
    err = float('inf')
    x_m = torch.max(x)
    eig_vec = torch.zeros((m.size()[0],))
    eig_val = 0
    while n < N_MAX and err >= VAL_ERR:
        x_u = x / x_m
        x = torch.matmul(m, x_u)

        eig_val = torch.max(x)
        eig_vec = x_u
        err = abs(eig_val - x_m)

        x_m = eig_val
        n += 1
        print(f'n: {n}, eig_vec: {eig_vec}, eig_val: {eig_val}')

    return eig_vec, eig_val

# 求 Householder 矩阵
def getHouseholderMatrix(vec):
    n = vec.size()[0]
    I = torch.eye(n)
    e1 = torch.zeros(n)
    e1[0] = 1
    v = vec + torch.norm(vec) * e1
    w = v / torch.norm(v)

    H = I - 2 * torch.matmul(w.view(-1, 1), w.view(1, -1))
    return H



def main(mat):
```

```
    n = mat.size()[0]
    eig_vec_list = []
    eig_val_list = []
    H_list = []
    r1_list = []
    evectmp_list = []
    for i in range(n):
        print(f'epoch: {i+1} ==========================================')
        eig_vec, eig_val = powerIteration(mat)
        eig_val_list.append(eig_val)

        if i == 0:
            eig_vec_list.append(eig_vec)
        else:
            evectmp_list.append(eig_vec)
            eval_idx = len(eig_val_list) - 1
            for H_mat, r1_vec, evectmp in zip(reversed(H_list), reversed(r1_list), reversed(evectmp_list)):
                alpha = r1_vec.dot(evectmp) / (eig_val_list[eval_idx] - eig_val_list[eval_idx - 1])
                alpha = alpha.unsqueeze(0)
                tmp_vec = torch.cat((alpha, evectmp)).view(-1, 1)
                final_vec = torch.matmul(H_mat, tmp_vec)

        H = getHouseholderMatrix(eig_vec) #  获得 Householder 矩阵
        A = torch.matmul(torch.matmul(H, mat), H.mT)    #  用 Householder 矩阵对原始矩阵做正交相似变换

        mat = A[1:, 1:]
        r1 = A[0, 1:]
        H_list.append(H)
        r1_list.append(r1)

        if mat.size()[0] == 1:
            eig_val_list.append(mat[0][0])
            break

    return torch.stack(eig_val_list).numpy(), eig_vec_list


if __name__ == '__main__':
    # mat = torch.rand((10, 10))
    mat = torch.tensor([[3, -1, 1], [2, 0, 1], [1, -1, 2]], dtype=torch.float32)
    print(f'mat: {mat}')
    eig_vals, eig_vecs = main(mat)
```

```
print(f'eig_vals={eig_vals}')

# test_evals, test_evecs = torch.eig(mat, eigenvectors=True)
# print(f'test_evals: {test_evals}')
# print(f'test_evects: {test_evecs}')

test2_evals, test2_evecs = np.linalg.eig(mat)
print(f'test2_evals: {test2_evals}')
print(f'test2_evecs: {test2_evecs}')
```