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

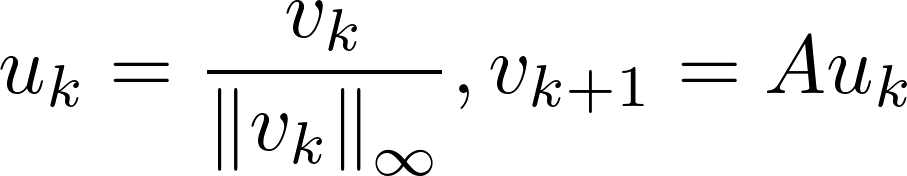
**实验目标**

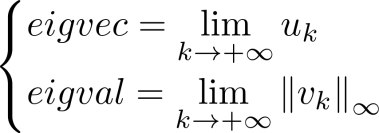
设计一种算法计算矩阵所有特征值与特征向量。

**实验步骤**

设有矩阵A为n阶方阵（wpsoffice），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：





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

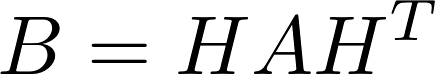
err = abs(eig\_val - x\_m)

x\_m = eig\_val

n += 1

return eig\_vec, eig\_val

1. 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：



1. 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:

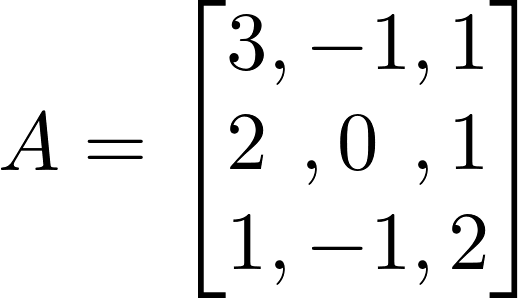
eig\_val\_list.append(mat[0][0])

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：



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：

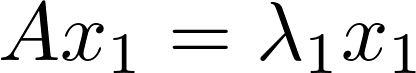
[2. 2. 1.]

可见在误差范围内两者几乎一样，算法是通过的。

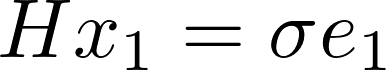
在计算出所有特征值后，可用消元法解线性方程组，得出所有特征向量，这里不做另外展示了。

**实验分析**

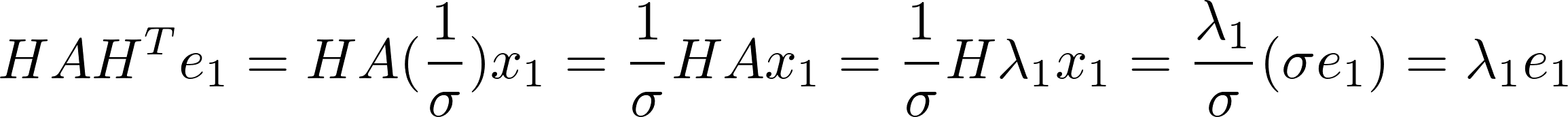
本实验使用收缩技术，假设已经求出了矩阵A的一个特征值wpsoffice及相应特征向量x1,即：



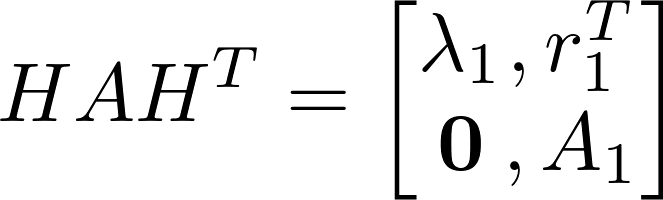
然后构造一个Householder变换矩阵，使得：

， wpsoffice

然后可得：



易知上式中 wpsoffice即为的第一列，所以有：



其中 wpsoffice, wpsoffice。

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 wpsoffice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}')