问题请教

问题

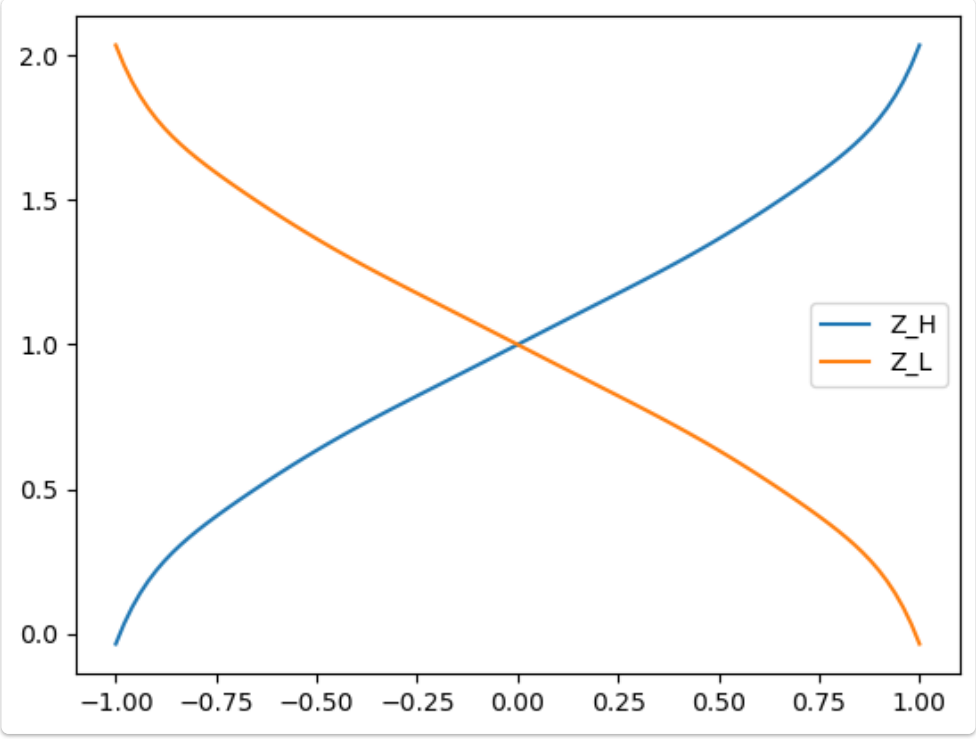
先验知识:$T_k(x) = \cos\left(k \arccos(x)\right)$,其中x在(-1,1)内.其迭代形式可写为:
$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$,其中$T_0(x) = 1.T_1(x) = x.$
给定切比雪夫光谱过滤器:$Z = \sum_{k=0}^{K} w_k T_k(\hat{\mathbf{L}})\mathbf{X}$.其中,$\hat{\mathbf{L}} = 2\tilde{\mathbf{L}}/\lambda_{max} - \hat{\mathbf{I}}$(可以当做$\tilde{A}$,范围在-1,1内).
重参数化[@heConvolutionalNeuralNetworks]参数$w_k = \frac{2}{K+1}\sum_{j=0}^{K}\gamma_j T_k(x_j)$,其中
$x_j = \cos\left(\frac{j+1/2}{K+1}\pi\right), j = 0, \ldots, K$表示切比雪夫点$T_{K+1}.T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$
那么为什么:当重参数[@chenPOLYGCLGRAPHCONTRASTIVE2024]
$\gamma_i^H = \sum_{j=0}^{i}\gamma_j, \quad \gamma_i^L = \gamma_0 - \sum_{j=1}^{i}\gamma_j, i = 1, \ldots, K$,后,
$\mathbf{Z}_L = f_\theta\left(\sum_{k=0}^{K} w_k^L T_k(\hat{\mathbf{L}})\mathbf{X}\right), \quad \mathbf{Z}_H = f_\theta\left(\sum_{k=0}^{K} w_k^H T_k(\hat{\mathbf{L}})\mathbf{X}\right)$分别代表低通滤波器和高通滤波器?
第一篇论文名:Convolutional Neural Networks on Graphs with Chebyshev Approximation, Revisited(chebnetII)
第二篇论文名:polygcl(其包含chebnetII)

部分解释以及附加实验:

解释1:γ_i^H中,高通滤波器的成分在不断地增加(即随着i的增加,γ_i^H在增加,对应$T_k(x_j)$的成分早增加).
γ_i^L与之同理.
疑问:γ_i^H的增加又不一定代表w_k^H的增加.令$Z = \sum_{k=0}^{K} w_k T_k(\mathbf{X})$.令X的范围为(-1,1).做如下**实验1**:
令$\gamma_0^H = 0, \gamma_0^L = 2, \gamma_i = \frac{2}{K}$.K=10(即多项式阶数).这样,我们可以得到$w_H$和$w_L$:

```
#$w_H$
tensor([ 2.0000e+00,  8.8857e-01, -2.1674e-08,  9.5838e-02, -1.0837e-07,
         3.2042e-02, -4.3349e-07,  1.3890e-02, -1.5172e-07,  5.5638e-03,
        -9.7535e-08])
#w_L
tensor([ 2.0000e+00, -8.8857e-01, -4.0640e-08, -9.5838e-02, -8.5004e-08,
        -3.2042e-02, -5.4186e-07, -1.3889e-02, -2.2216e-07, -5.5641e-03,
         1.5308e-07])
```

这可以得到如下的结论:T_k的系数并不是递减的.用如上解释1不太合理.因此,做如下额外的**实验2**:
将其代入Z,并令X=1后,可以得到:($Z_H(1)$ =tensor(2.0359), $Z_H(1)$ =tensor(-0.0359))
做额外的如下实验:
实验3
由于x的范围是(-1,1),剩余设置如实验1所示.画出Z_H和Z_L随着X的变动的图如下:



γ_L和γ_H

γ_i

Z_H

Z_L

附录(相关代码):

utils:

```
from torch_geometric.nn import MessagePassing
from torch_geometric.utils import get_laplacian
import torch.nn as nn
import torch
import torch.nn.functional as F

def get_cheb_i(x, i):
    if i == 0:
        return 1
    elif i == 1:
        return x
    else:
        T0 = 1
        T1 = x
        for ii in range(2, i + 1):
            T2 = 2 * x * T1 - T0
```

```
            T0, T1 = T1, T2
        return T2
def prefix_sum(gammas,gamma_0):
    gammas_H=torch.cat([gamma_0,gammas],dim=0)
    #cumulative sum:累计求和,这个真好用
    return torch.cumsum(gammas_H,dim=0)
def prefix_diff(gammas,gamma_0):
    gammas_L=torch.cat([gamma_0,-gammas],dim=0)
    return torch.cumsum(gammas_L,dim=0)
def get_ws(prefixed_gammas,x_j,K):
    ws=prefixed_gammas.clone()
    ws.fill_(0)
    for k in range(0,K+1):
        for j in range(0,K+1):
            ws[k]=ws[k]+prefixed_gammas[j]*get_cheb_i(x_j[j],k)
        ws[k]=2*ws[k]/(K+1)
    return ws
def get_poly(x,K,gamma_H,gamma_L):
    x_j = torch.Tensor([(j + 0.5) / (K + 1) * torch.pi for j in range(0, K + 1)]
[::-1])
    print(x_j)
    x_j = torch.cos(x_j)
    print(x_j)
    gammas=torch.Tensor(K)
    gammas.fill_(2/K)#default 2
    gamma_H=torch.Tensor([gamma_H])
    gamma_L=torch.Tensor([gamma_L])
    prefix_gammas_H=prefix_sum(F.relu(gammas),gamma_H)
    prefix_gammas_L=prefix_diff(F.relu(gammas),gamma_L)
    prefix_gammas_L=F.relu(prefix_gammas_L)
    ws_H=get_ws(prefix_gammas_H,x_j,K)
    ws_L=get_ws(prefix_gammas_L,x_j,K)
    #实验二的部分,输入w_H和w_L
    print(ws_H)
    print(ws_L)
    out_H=ws_H[0]/2
    out_L=ws_L[0]/2
    for k in range(1,K+1):
        out_H=out_H+ws_H[k]*get_cheb_i(x,k)
        out_L=out_L+ws_L[k]*get_cheb_i(x,k)
    return out_H,out_L
```

实验一,实验二

```
get_poly(1,10,0,2)
```

实验三

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(-1, 1, 100)
y=[]
y_H,y_L=get_poly(x,K=10,gamma_H=0,gamma_L=2)
plt.plot(x,y_H,label='Z_H')
plt.plot(x,y_L,label='Z_L')
plt.legend()
```