Python小练习：向量之间的距离度量

作者：凯鲁嘎吉 - 博客园 http://www.cnblogs.com/kailugaji/

本文主要用Python实现三种常见的向量之间的距离度量方式：

1）曼哈顿距离(Manhattan distance, L1范数)：$d(x,y) = \sum\limits_{i = 1}^n {\left| {{x_i} - {y_i}} \right|} $

2）欧氏距离(Euclidean distance，L2范数)：$d(x,y) = \sqrt {\sum\limits_{i = 1}^n {{{({x_i} - {y_i})}^2}} } $

3）余弦相似度(Cosine similarity)：$d(x,y) = \frac{{x{y^T}}}{{\left\| x \right\|\left\| y \right\|}}$

其中，$x,y \in \mathbb{R}{^{1 \times n}}$

1. loss_test.py

```python
# -*- coding: utf-8 -*-
# Author：凯鲁嘎吉 Coral Gajic
# https://www.cnblogs.com/kailugaji/
# Python小练习：向量之间的距离度量
# Python实现两向量之间的：
# 1）曼哈顿距离(Manhattan distance, L1范数)
# 2）欧氏距离(Euclidean distance，L2范数)
# 3）余弦相似度(Cosine similarity)
import torch
import torch.nn.functional as F
# 自己写的距离度量函数
def compute_l1_similarity(e1, e2): # L1距离
    return torch.abs(e1 - e2).sum(-1)
def compute_l2_similarity(e1, e2): # L2距离
    return ((e1 - e2)**2).sum(-1).sqrt()
    # 注意：这里开根号了，没平方
def compute_cosine_similarity(e1, e2): # cosine距离
    e1 = e1 / torch.norm(e1, dim=-1, p=2, keepdim=True)
    e2 = e2 / torch.norm(e2, dim=-1, p=2, keepdim=True)
    similarity = torch.mul(e1, e2).sum(1) # mul: 点乘
    return similarity
    # 后两行也可替换为：
    # similarity = torch.mm(e1, torch.t(e2)) # mm: 相乘，torch.t: 转置
```

```
    # return torch.diag(similarity) # 只取对角线元素

torch.manual_seed(1)
n = 3 # 样本个数
m = 5 # 样本维度
# 仅考虑e1的第i个样本和e2的第i个样本之间计算距离
# 不考虑e1的i个样本和e2的第j个样本之间的距离(i≠j)
e1 = torch.randn(n, m)
e2 = torch.randn(n, m)
print('原始数据为：\n', e1, '\n', e2)
loss_l1_1 = torch.zeros(n)
loss_l2_1 = torch.zeros(n)
# 自己写的距离度量函数
loss_l1 = compute_l1_similarity(e1, e2)
loss_l2 = compute_l2_similarity(e1, e2)
loss_cosine = compute_cosine_similarity(e1, e2)
# pytorch库里自带的距离度量函数
for i in range(n):
    loss_l1_1[i] = torch.dist(e1[i], e2[i], p=1)
    loss_l2_1[i] = torch.dist(e1[i], e2[i], p=2)
loss_cosine_1 = F.cosine_similarity(e1, e2)
# 第一个结果是自己写的函数
# 第二个结果是pytorch库里自带的函数
# n是多少，就出来多少个值
print('两者的曼哈顿距离为：\n', loss_l1, '\n', loss_l1_1)
print('两者的欧式距离为：\n', loss_l2, '\n', loss_l2_1)
print('两者的余弦相似度为：\n', loss_cosine, '\n', loss_cosine_1)
```

2. 结果

```
D:\ProgramData\Anaconda3\python.exe "D:/Python code/2023.3 exercise/loss/loss_test.py"
原始数据为：
 tensor([[ 0.6614,  0.2669,  0.0617,  0.6213, -0.4519],
        [-0.1661, -1.5228,  0.3817, -1.0276, -0.5631],
        [-0.8923, -0.0583, -0.1955, -0.9656,  0.4224]])
 tensor([[ 0.2673, -0.4212, -0.5107, -1.5727, -0.1232],
        [ 3.5870, -1.8313,  1.5987, -1.2770,  0.3255],
        [-0.4791,  1.3790,  2.5286,  0.4107, -0.9880]])
两者的曼哈顿距离为：
 tensor([4.1772, 6.4166, 7.3613])
 tensor([4.1772, 6.4166, 7.3613])
两者的欧式距离为：
 tensor([2.4245, 4.0637, 3.6798])
 tensor([2.4245, 4.0637, 3.6798])
两者的余弦相似度为：
```

```
 tensor([-0.4887,  0.4416, -0.2214])
 tensor([-0.4887,  0.4416, -0.2214])

Process finished with exit code 0
```

注意：这里只是求向量之间的距离度量，并不是矩阵范数。上下两个结果分别为自己根据距离定义写的函数、pytorch自带的函数，可以看到得到的结果是一致的。

余弦相似性值越大两向量越相似，曼哈顿距离与欧式距离值越小两向量越相似。

补充：将距离度量应用在聚类划分上。给定原始数据与训练好的聚类中心，根据上述几种距离度量指标，计算原始数据与聚类中心之间的相似度，依照相似度值来划分数据，判断每一个样本属于哪一个类别，并得到每一个样本的类标签。

```python
# -*- coding: utf-8 -*-
# Author：凯鲁嘎吉 Coral Gajic
# https://www.cnblogs.com/kailugaji/
# 已知数据与聚类中心，根据距离度量指标将数据划分为不同的类
import torch

def compute_l1_similarity(e1, e2):  # L1距离
    return torch.abs(e1 - e2).sum(-1)

def compute_l2_similarity(e1, e2):  # L2距离
    return ((e1 - e2) ** 2).sum(-1).sqrt()
    # 注意：这里开根号了，没平方

def compute_cosine_similarity(e1, e2):  # cosine距离
    e1 = e1 / torch.norm(e1, dim=-1, p=2, keepdim=True)
    e2 = e2 / torch.norm(e2, dim=-1, p=2, keepdim=True)
    similarity = torch.mul(e1, e2).sum()  # mul: 点乘
    return similarity

def compute_label(data, center, method):
    global label
    z_sim = torch.zeros([len(data), len(center)])
    for i in range(len(data)):
        for j in range(len(center)):
            if method == 'L1':
                z_sim[i][j] = compute_l1_similarity(data[i], center[j])
                label = z_sim.argmin(dim=1).numpy()
            elif method == 'L2':
                z_sim[i][j] = compute_l2_similarity(data[i], center[j])
                label = z_sim.argmin(dim=1).numpy()
            elif method == 'cosine':
```

```
                z_sim[i][j] = compute_cosine_similarity(data[i], center[j])
                label = z_sim.argmax(dim=1).numpy()
    print('%s相似性：\n' % method, z_sim)
    print('样本类别标签：', label)

# 样本
data = (torch.tensor([[1, 1, 1, 1, 1],
                      [-1, -2, -3, -4, -5],
                      [2, 4, 6, 8, 11],
                      [-1, -1, -1, 1, -1],
                      [10, 9, 8, 6, 5],
                      [-7, -3, -2, 4, -5],
                      [3, -2, 5, 8, 5]])) * 1.0
# 聚类中心
center = (torch.tensor([[2, 2, 6, 2, 2],
                        [-1, -1, -1, 5, -1],
                        [1, -1, 3, 4, 5]])) * 1.0
print('样本： \n', data)
print('聚类中心： \n', center)
print('-------------------------------------')
compute_label(data, center, 'L1')
print('-------------------------------------')
compute_label(data, center, 'L2')
print('-------------------------------------')
compute_label(data, center, 'cosine')
```

结果:

```
D:\ProgramData\Anaconda3\python.exe "D:/Python code/2023.3 exercise/向量间的距离度量/test.py"
样本：
 tensor([[ 1.,  1.,  1.,  1.,  1.],
        [-1., -2., -3., -4., -5.],
        [ 2.,  4.,  6.,  8., 11.],
        [-1., -1., -1.,  1., -1.],
        [10.,  9.,  8.,  6.,  5.],
        [-7., -3., -2.,  4., -5.],
        [ 3., -2.,  5.,  8.,  5.]])
聚类中心：
 tensor([[ 2.,  2.,  6.,  2.,  2.],
        [-1., -1., -1.,  5., -1.],
        [ 1., -1.,  3.,  4.,  5.]])
-------------------------------------
L1相似性：
 tensor([[ 9., 12., 11.],
        [29., 16., 27.],
        [17., 30., 19.],
        [17.,  4., 15.],
```

```
        [24., 37., 26.],
        [31., 14., 25.],
        [15., 20.,  9.]])
样本类别标签： [0 1 0 1 0 1 2]
-----------------------------------
L2相似性：
 tensor([[ 5.3852,  5.6569,  5.7446],
        [13.8203, 10.0995, 14.3178],
        [11.0000, 15.3623,  9.3274],
        [ 8.7750,  4.0000,  8.0623],
        [11.9164, 18.4120, 14.4914],
        [14.9332,  7.6158, 13.8924],
        [ 7.9373,  9.8995,  5.0000]])
样本类别标签： [0 1 2 1 0 1 2]
-----------------------------------
cosine相似性：
 tensor([[ 0.8682,  0.0830,  0.7442],
        [-0.7854, -0.2254, -0.9162],
        [ 0.7682,  0.2033,  0.9201],
        [-0.6202,  0.7474, -0.2481],
        [ 0.8562, -0.0212,  0.5866],
        [-0.4646,  0.6770, -0.2596],
        [ 0.7137,  0.4779,  0.9475]])
样本类别标签： [0 1 2 1 0 1 2]

Process finished with exit code 0
```

注意：data为原始数据，一共有7个样本，每个样本维度都为5，center为聚类中心，有3个聚类中心(类别)。

3. 参考文献

[1] 相似性度量 - 凯鲁嘎吉 - 博客园

[2] 向量范数与矩阵范数 - 凯鲁嘎吉 - 博客园