PyTorch线性代数

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

所用版本：python 3.6.5，torch 1.6.0，torchvision 0.7.0

1. 标量运算

```
In [1]: import torch
```

```
In [2]: x = torch.tensor([3.0])
```

```
In [3]: y = torch.tensor([2.0])
```

```
In [4]: x + y, x * y, x / y, x**y
Out[4]: (tensor([5.]), tensor([6.]), tensor([1.5000]), tensor([9.]))
```

2. 向量运算

```
In [5]: x = torch.arange(12)
```

In [6]:
```
x
```

Out[6]:
```
tensor([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

In [7]:
```
x[3]
```

Out[7]:
```
tensor(3)
```

In [8]:
```
len(x)
```

Out[8]:
```
12
```

In [9]:
```
x.shape
```

Out[9]:
```
torch.Size([12])
```

In [10]:
```
y = torch.ones(12, dtype=torch.float32)
```

In [11]:
```
y
```

Out[11]:
```
tensor([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

转换数据类型

In [12]:
```
x = x.float()
```

In [13]:
```
x
```

Out[13]:
```
tensor([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10., 11.])
```

点积：给定两个向量$\mathbf{x},\mathbf{y}\in\mathbb{R}^d$，它们的*点积*（dotproduct）$\mathbf{x}^\top\mathbf{y}$（或$\langle\mathbf{x},\mathbf{y}\rangle$）是相同位置的按元素乘积的和：$\mathbf{x}^\top \mathbf{y} = \sum_{i=1}^{d} x_i y_i$。

In [14]:
```
torch.dot(x, y)
```

Out[14]:
```
tensor(66.)
```

In [15]:
```
torch.sum(x * y)
```

Out[15]:
```
tensor(66.)
```

向量所有元素相乘

In [16]:
```
torch.prod(x)
```

Out[16]:
```
tensor(0.)
```

In [17]:
```
torch.prod(y)
```

Out[17]:
```
tensor(1.)
```

In [18]:
```
x + y
```

Out[18]:
```
tensor([ 1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10., 11., 12.])
```

向量所有元素相加

In [19]:
```
torch.sum(x)
```

Out[19]:
```
tensor(66.)
```

In [20]:
```
torch.sum(y)
```

Out[20]:
```
tensor(12.)
```

求均值

```
In  [21]: x.mean()
```

```
Out[21]: tensor(5.5000)
```

```
In  [22]: x.numel()
```

```
Out[22]: 12
```

```
In  [23]: x_size = float(x.numel())
```

```
In  [24]: x.sum() / x_size
```

```
Out[24]: tensor(5.5000)
```

将向量转化为矩阵

```
In  [25]: x.reshape(3, 4)
```

```
Out[25]: tensor([[ 0.,  1.,  2.,  3.],
                 [ 4.,  5.,  6.,  7.],
                 [ 8.,  9., 10., 11.]])
```

3. 张量运算

```
In  [26]: X = torch.arange(12, dtype=torch.float32).reshape((3,4))
```

```
In  [27]: X
```

```
Out[27]: tensor([[ 0.,  1.,  2.,  3.],
                 [ 4.,  5.,  6.,  7.],
                 [ 8.,  9., 10., 11.]])
```

axis=0时，返回矩阵X每一列最大元素所在下标

```
In  [28]: torch.argmax(X, dim = 0)
```

```
Out[28]: tensor([2, 2, 2, 2])
```

axis=1时，返回矩阵X每一行最大元素所在下标

```
In  [29]: torch.argmax(X, dim = 1)
```

```
Out[29]: tensor([3, 3, 3])
```

axis=0时，返回矩阵X每一列求和结果

```
In  [30]: X.sum(axis = 0)
```

```
Out[30]: tensor([12., 15., 18., 21.])
```

axis=1时，返回矩阵X每一行求和结果

```
In  [31]: X.sum(axis = 1)
```

```
Out[31]: tensor([ 6., 22., 38.])
```

axis=[0, 1]，先对列求和，再对行求和，即矩阵所有元素相加的结果

```
In  [32]: X.sum(axis = [0, 1])
Out[32]: tensor(66.)

In  [33]: X.sum()
Out[33]: tensor(66.)

In  [34]: Y = torch.tensor([[2.0, 1, 4, 3], [1, 2, 3, 4], [4, 3, 2, 1]])

In  [35]: Y
Out[35]: tensor([[2., 1., 4., 3.],
                 [1., 2., 3., 4.],
                 [4., 3., 2., 1.]])

In  [36]: torch.argmax(Y, dim = 0)
Out[36]: tensor([2, 2, 0, 1])

In  [37]: torch.argmax(Y, dim = 1)
Out[37]: tensor([2, 3, 0])
```

axis=0时，X与Y按行连接

In [38]:
```
torch.cat((X, Y), dim=0)
```

Out[38]:
```
tensor([[ 0.,  1.,  2.,  3.],
        [ 4.,  5.,  6.,  7.],
        [ 8.,  9., 10., 11.],
        [ 2.,  1.,  4.,  3.],
        [ 1.,  2.,  3.,  4.],
        [ 4.,  3.,  2.,  1.]])
```

axis=1时，X与Y按列连接

In [39]:
```
torch.cat((X, Y), dim=1)
```

Out[39]:
```
tensor([[ 0.,  1.,  2.,  3.,  2.,  1.,  4.,  3.],
        [ 4.,  5.,  6.,  7.,  1.,  2.,  3.,  4.],
        [ 8.,  9., 10., 11.,  4.,  3.,  2.,  1.]])
```

矩阵对应元素相加

In [40]:
```
X + Y
```

Out[40]:
```
tensor([[ 2.,  2.,  6.,  6.],
        [ 5.,  7.,  9., 11.],
        [12., 12., 12., 12.]])
```

矩阵的转置

In [41]:
```
Z = X.T
```

In [42]:
```
Z
```

Out[42]:
```
tensor([[ 0.,  4.,  8.],
        [ 1.,  5.,  9.],
        [ 2.,  6., 10.],
        [ 3.,  7., 11.]])
```

矩阵对应元素相乘

In [43]:
```
X * Y
```

Out[43]:
```
tensor([[ 0.,  1.,  8.,  9.],
        [ 4., 10., 18., 28.],
        [32., 27., 20., 11.]])
```

矩阵相乘 A=Z*Z'

In [44]:
```
A = torch.mm(Z, Z.T)
```

In [45]:
```
A
```

Out[45]:
```
tensor([[ 80.,  92., 104., 116.],
        [ 92., 107., 122., 137.],
        [104., 122., 140., 158.],
        [116., 137., 158., 179.]])
```

构建对称矩阵，A_symm=(A+A')/2

In [46]:
```
A_symm = (A + A.T) / 2.0
```

In [47]:
```
A_symm
```

Out[47]:
```
tensor([[ 80.,  92., 104., 116.],
        [ 92., 107., 122., 137.],
        [104., 122., 140., 158.],
        [116., 137., 158., 179.]])
```

判断A_symm是否为对称阵，即A_symm=A_symm'

In [48]:
```
A_symm == A_symm.T
```

Out[48]:
```
tensor([[True, True, True, True],
        [True, True, True, True],
        [True, True, True, True],
        [True, True, True, True]])
```

计算总和或均值时保持轴数不变

In [49]:
```
sum_X = X.sum(axis=1, keepdims=True)
```

In [50]:
```
sum_X
```

Out[50]:
```
tensor([[ 6.],
        [22.],
        [38.]])
```

由于sum_X在对每行进行求和后仍保持两个轴，我们可以通过广播将X除以sum_X。

In [51]:
```
X / sum_X
```

Out[51]:
```
tensor([[0.0000, 0.1667, 0.3333, 0.5000],
        [0.1818, 0.2273, 0.2727, 0.3182],
        [0.2105, 0.2368, 0.2632, 0.2895]])
```

沿某个轴计算X元素的累积总和，比如axis=0（按行计算），我们可以调用cumsum函数。此函数不会沿任何轴降低输入张量的维度。

In [52]:
```
X.cumsum(axis=0)
```

Out[52]:
```
tensor([[ 0.,  1.,  2.,  3.],
        [ 4.,  6.,  8., 10.],
        [12., 15., 18., 21.]])
```

In [53]:
```
X.cumsum(axis=1)
```

Out[53]:
```
tensor([[ 0.,  1.,  3.,  6.],
        [ 4.,  9., 15., 22.],
        [ 8., 17., 27., 38.]])
```

In [54]:
```
Z = torch.arange(24).reshape(2, 3, 4)
```

In [55]:
```
Z
```

Out[55]:
```
tensor([[[ 0,  1,  2,  3],
         [ 4,  5,  6,  7],
         [ 8,  9, 10, 11]],

        [[12, 13, 14, 15],
         [16, 17, 18, 19],
         [20, 21, 22, 23]]])
```

标量乘以张量

In [56]:
```
a = 2
```

In [57]:
```
a + Z
```

Out[57]:
```
tensor([[[ 2,  3,  4,  5],
         [ 6,  7,  8,  9],
         [10, 11, 12, 13]],

        [[14, 15, 16, 17],
         [18, 19, 20, 21],
         [22, 23, 24, 25]]])
```

In [58]:
```
a * Z
```

Out[58]:
```
tensor([[[ 0,  2,  4,  6],
         [ 8, 10, 12, 14],
         [16, 18, 20, 22]],

        [[24, 26, 28, 30],
         [32, 34, 36, 38],
         [40, 42, 44, 46]]])
```

矩阵乘以向量 $$ \mathbf{X}\mathbf{b} = \begin{bmatrix} \mathbf{x}^\top_{1} \\ \mathbf{x}^\top_{2} \\ \vdots \\ \mathbf{x}^\top_m \\ \end{bmatrix}\mathbf{b} = \begin{bmatrix} \mathbf{x}^\top_{1} \mathbf{b} \\ \mathbf{x}^\top_{2} \mathbf{b} \\ \vdots\\ \mathbf{x}^\top_{m} \mathbf{b}\\ \end{bmatrix}. $$

In [59]:
```
b = torch.tensor([2.0, 1, 4, 3])
```

In [60]:
```
b
```

Out[60]:
```
tensor([2., 1., 4., 3.])
```

In [61]:
```
X
```

Out[61]:
```
tensor([[ 0.,  1.,  2.,  3.],
        [ 4.,  5.,  6.,  7.],
        [ 8.,  9., 10., 11.]])
```

In [62]:
```
torch.mv(X, b)
```

Out[62]:
```
tensor([18., 58., 98.])
```

4. 范数

2范数
$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^n x_i^2},$$

In [63]:
```
u = torch.tensor([3.0, -4.0])
```

In [64]:
```
torch.norm(u, p=2)
```

Out[64]:
```
tensor(5.)
```

1范数
$$\|\mathbf{x}\|_1 = \sum_{i=1}^n \left|x_i \right|.$$

In [65]:
```
torch.abs(u).sum()
```

Out[65]:
```
tensor(7.)
```

In [66]:
```
torch.norm(u, p=1)
```

Out[66]:
```
tensor(7.)
```

$\infty $范数
$$\|\mathbf{x}\|_\infty = max(|x_{i}|).$$

In [67]:
```
import numpy as np
```

In [68]:
```
torch.norm(u, p=np.inf)
```

Out[68]:
```
tensor(4.)
```