TensorFlow线性代数

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

1. 标量运算

In [1]:
```
import tensorflow as tf
```

In [2]:
```
x = tf.constant([3.0])
```

In [3]:
```
y = tf.constant([2.0])
```

In [4]:
```
sess = tf.Session()
```

In [5]:
```
sess.run([x + y, x * y, x / y, x**y])
```

Out[5]:
```
[array([5.], dtype=float32),
 array([6.], dtype=float32),
 array([1.5], dtype=float32),
 array([9.], dtype=float32)]
```

2. 向量运算

In [6]:
```
x = tf.range(12)
```

In [7]:
```
sess.run(x)
```

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

In [8]:
```
sess.run(x[3])
```

Out[8]:
```
3
```

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

Out[9]:
```
TensorShape([Dimension(12)])
```

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

In [11]:
```
sess.run(y)
```

Out[11]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.], dtype=float32)
```

转换数据类型

In [12]:
```
x = tf.cast(x, "float32")
```

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 [13]:
```
sess.run(tf.tensordot(x, y, axes=1))
```

Out[13]:
```
66.0
```

In [14]:
```
sess.run(tf.reduce_sum(x * y))
```

Out[14]:
```
66.0
```

向量所有元素相乘

```
In  [15]: sess.run(tf.reduce_prod(x))
```

```
Out[15]: 0.0
```

```
In  [16]: sess.run(tf.reduce_prod(y))
```

```
Out[16]: 1.0
```

```
In  [17]: sess.run(x + y)
```

```
Out[17]: array([ 1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10., 11., 12.],
               dtype=float32)
```

向量所有元素相加

```
In  [18]: sess.run(tf.reduce_sum(x))
```

```
Out[18]: 66.0
```

```
In  [19]: sess.run(tf.reduce_sum(y))
```

```
Out[19]: 12.0
```

求均值

```
In  [20]: sess.run(tf.reduce_mean(x))
```

```
Out[20]: 5.5
```

In [21]:
```
sess.run(tf.size(x))
```

Out[21]: 12

In [22]:
```
x_size = tf.cast(tf.size(x), "float32")
```

In [23]:
```
sess.run(tf.reduce_sum(x) / x_size)
```

Out[23]: 5.5

将向量转化为矩阵

In [24]:
```
sess.run(tf.reshape(x, (3, 4)))
```

Out[24]:
```
array([[ 0.,  1.,  2.,  3.],
       [ 4.,  5.,  6.,  7.],
       [ 8.,  9., 10., 11.]], dtype=float32)
```

3. 矩阵/张量运算

In [25]:
```
X = tf.reshape(tf.range(12, dtype=tf.float32), (3, 4))
```

In [26]:
```
sess.run(X)
```

Out[26]:
```
array([[ 0.,  1.,  2.,  3.],
       [ 4.,  5.,  6.,  7.],
       [ 8.,  9., 10., 11.]], dtype=float32)
```

axis=0时，返回矩阵X每一列最大元素所在下标

In [27]:
```
sess.run(tf.argmax(X, 0))
```

Out[27]: array([2, 2, 2, 2], dtype=int64)

axis=1时，返回矩阵X每一行最大元素所在下标

In [28]:
```
sess.run(tf.argmax(X, 1))
```

Out[28]: array([3, 3, 3], dtype=int64)

axis=0时，返回矩阵X每一列求和结果

In [29]:
```
sess.run(tf.reduce_sum(X, axis=0))
```

Out[29]: array([12., 15., 18., 21.], dtype=float32)

axis=1时，返回矩阵X每一行求和结果

In [30]:
```
sess.run(tf.reduce_sum(X, axis=1))
```

Out[30]: array([6., 22., 38.], dtype=float32)

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 [31]:
```
sess.run(tf.reduce_sum(X, axis=[0, 1]))
```

Out[31]: 66.0

```
In  [32]: sess.run(tf.reduce_sum(X))
Out[32]: 66.0

In  [33]: Y = tf.constant([[2.0, 1, 4, 3], [1, 2, 3, 4], [4, 3, 2, 1]])

In  [34]: sess.run(Y)
Out[34]: array([[2., 1., 4., 3.],
                [1., 2., 3., 4.],
                [4., 3., 2., 1.]], dtype=float32)

In  [35]: sess.run(tf.argmax(Y, 0))
Out[35]: array([2, 2, 0, 1], dtype=int64)

In  [36]: sess.run(tf.argmax(Y, 1))
Out[36]: array([2, 3, 0], dtype=int64)
```

axis=0时，X与Y按行连接

```
In  [37]: sess.run(tf.concat([X, Y], axis=0))
Out[37]: array([[ 0.,  1.,  2.,  3.],
                [ 4.,  5.,  6.,  7.],
                [ 8.,  9., 10., 11.],
                [ 2.,  1.,  4.,  3.],
                [ 1.,  2.,  3.,  4.],
                [ 4.,  3.,  2.,  1.]], dtype=float32)
```

axis=1时，X与Y按列连接

```
In  [38]: sess.run(tf.concat([X, Y], axis=1))

Out[38]: array([[ 0.,  1.,  2.,  3.,  2.,  1.,  4.,  3.],
                [ 4.,  5.,  6.,  7.,  1.,  2.,  3.,  4.],
                [ 8.,  9., 10., 11.,  4.,  3.,  2.,  1.]], dtype=float32)
```

矩阵对应元素相加

```
In  [39]: sess.run(X + Y)

Out[39]: array([[ 2.,  2.,  6.,  6.],
                [ 5.,  7.,  9., 11.],
                [12., 12., 12., 12.]], dtype=float32)
```

矩阵的转置

```
In  [40]: Z = tf.transpose(X)

In  [41]: sess.run(Z)

Out[41]: array([[ 0.,  4.,  8.],
                [ 1.,  5.,  9.],
                [ 2.,  6., 10.],
                [ 3.,  7., 11.]], dtype=float32)
```

矩阵对应元素相乘

```
In  [42]: sess.run(X * Y)

Out[42]: array([[ 0.,  1.,  8.,  9.],
                [ 4., 10., 18., 28.],
                [32., 27., 20., 11.]], dtype=float32)
```

矩阵相乘 A=Z*Z'

```
In  [43]: A = tf.matmul(Z, tf.transpose(Z))

In  [44]: sess.run(A)

Out[44]: array([[ 80.,  92., 104., 116.],
                [ 92., 107., 122., 137.],
                [104., 122., 140., 158.],
                [116., 137., 158., 179.]], dtype=float32)
```

构建对称矩阵，A_symm=(A+A')/2

```
In  [45]: A_symm = (A + tf.transpose(A)) / 2.0

In  [46]: sess.run(A_symm)

Out[46]: array([[ 80.,  92., 104., 116.],
                [ 92., 107., 122., 137.],
                [104., 122., 140., 158.],
                [116., 137., 158., 179.]], dtype=float32)
```

判断A_symm是否为对称阵，即A_symm=A_symm'

```
In  [47]: sess.run(tf.equal(A_symm, tf.transpose(A_symm)))

Out[47]: array([[ 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 [48]:
```
sum_X = tf.reduce_sum(X, axis=1, keepdims=True)
```

In [49]:
```
sess.run(sum_X)
```

Out[49]:
```
array([[ 6.],
       [22.],
       [38.]], dtype=float32)
```

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 [50]:
```
sess.run(X / sum_X)
```

Out[50]:
```
array([[0.        , 0.16666667, 0.33333334, 0.5       ],
       [0.18181819, 0.22727273, 0.27272728, 0.3181818 ],
       [0.21052632, 0.23684211, 0.2631579 , 0.28947368]], dtype=float32)
```

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 [51]:
```
sess.run(tf.cumsum(X, axis=0))
```

Out[51]:
```
array([[ 0.,  1.,  2.,  3.],
       [ 4.,  6.,  8., 10.],
       [12., 15., 18., 21.]], dtype=float32)
```

In [52]:
```
sess.run(tf.cumsum(X, axis=1))
```

Out[52]:
```
array([[ 0.,  1.,  3.,  6.],
       [ 4.,  9., 15., 22.],
       [ 8., 17., 27., 38.]], dtype=float32)
```

In [53]:
```
Z = tf.reshape(tf.range(24), (2, 3, 4))
```

In [54]:
```
sess.run(Z)
```

Out[54]:
```
array([[[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]],

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

标量乘以矩阵

In [55]:
```
a = 2
```

In [56]:
```
sess.run(a + Z)
```

Out[56]:
```
array([[[ 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 [57]:
```
sess.run(a * Z)
```

Out[57]:
```
array([[[ 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  [58]: b = tf.constant([2.0, 1, 4, 3])

In  [59]: sess.run(b)
Out[59]: array([2., 1., 4., 3.], dtype=float32)

In  [60]: sess.run(X)
Out[60]: array([[ 0.,  1.,  2.,  3.],
                [ 4.,  5.,  6.,  7.],
                [ 8.,  9., 10., 11.]], dtype=float32)
```

把向量b扩展成与矩阵X大小一致

```
In  [61]: b = tf.expand_dims(b, 1)

In  [62]: sess.run(b)
Out[62]: array([[2.],
                [1.],
                [4.],
                [3.]], dtype=float32)

In  [63]: sess.run(tf.matmul(X, b))
Out[63]: array([[18.],
                [58.],
                [98.]], dtype=float32)
```

4. 范数

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

In [64]:
```
u = tf.constant([3.0, -4.0])
```

In [65]:
```
sess.run(tf.norm(u, ord=2))
```

Out[65]: 5.0

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

In [66]:
```
sess.run(tf.reduce_sum(tf.abs(u)))
```

Out[66]: 7.0

In [67]:
```
sess.run(tf.norm(u, ord=1))
```

Out[67]: 7.0

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

In [68]:
```
import numpy as np
```

In [69]:
```
sess.run(tf.norm(u, ord=np.inf))
```

Out[69]: 4.0