tensorflow接口研读math_ops(二)

math_ops函数使用，本篇为线性代数，复数和fft。

1.51 tf.diag(diagonal, name=None)

功能：返回对角阵。
输入：tensor，秩为k<=3。
例：

```
a=tf.constant([1,2,3,4])
z=tf.diag(a)

z==>[[1 0 0 0
      0 2 0 0
      0 0 3 0
      0 0 0 4]]
```

1.52 tf.diag_part(input,name=None)

功能：返回对角阵的对角元素。
输入：tensor,且维度必须一致。
例：

```
a=tf.constant([[1,5,0,0],[0,2,0,0],[0,0,3,0],[0,0,0,4]])
z=tf.diag_part(a)

z==>[1,2,3,4]
```

1.53 tf.trace(x,name=None)

功能：返回矩阵的迹。
输入：tensor
例：

```
a=tf.constant([[[1,2,3],[4,5,6],[7,8,9]],[[10,11,12],[13,14,15],[16,17,18]]])
z=tf.trace(a)

z==>[15 42]
```

1.54 tf.transpose(a,perm=None,name='transpose')

功能：矩阵转置。
输入：tensor，perm代表转置后的维度排列，决定了转置方法，默认为[n-1,....,0]，n为a的维度。
例：

```
a=tf.constant([[1,2,3],[4,5,6]])
z=tf.transpose(a)#perm为[1,0]，即0维和1维互换。

z==>[[1 4]
     [2 5]
     [3 6]]
```

1.55 tf.eye(num_rows, num_columns=None, batch_shape=None, dtype=tf.float32, name=None)

功能：返回单位阵。
输入：num_rows:矩阵的行数;num_columns:矩阵的列数，默认与行数相等
batch_shape:若提供值，则返回batch_shape的单位阵。
例：

```
z=tf.eye(2,batch_shape=[2])

z==>[[[1. 0.]
      [0. 1.]]
     [[1. 0.]
      [0. 1.]]]
```

1.56 tf.matrix_diag(diagonal,name=None)

功能：根据对角值返回一批对角阵
输入：对角值
例：

```
a=tf.constant([[1,2,3],[4,5,6]])
z=tf.matrix_diag(a)

z==>[[[1 0 0]
      [0 2 0]
      [0 0 3]]
```

```
     [[4 0 0]
      [0 5 0]
      [0 0 6]]]
```

1.57 tf.matrix_diag_part(input,name=None)

功能：返回批对角阵的对角元素
输入：tensor,批对角阵
例：

```
a=tf.constant([[[1,3,0],[0,2,0],[0,0,3]],[[4,0,0],[0,5,0],[0,0,6]]])
z=tf.matrix_diag_part(a)

z==>[[1 2 3]
     [4 5 6]]
```

1.58 tf.matrix_band_part(input,num_lower,num_upper,name=None)

功能：复制一个矩阵，并将规定带之外的元素置为0。
假设元素坐标为（m，n），则in_band(m, n) = (num_lower < 0 || (m-n) <= num_lower)) &&
(num_upper < 0 || (n-m) <= num_upper)。
band（m,n）=in_band(m,n)*input(m,n)。
特殊情况：
tf.matrix_band_part(input, 0, -1) ==> 上三角阵.
tf.matrix_band_part(input, -1, 0) ==> 下三角阵.
tf.matrix_band_part(input, 0, 0) ==> 对角阵.
输入：num_lower:如果为负，则结果右上空三角阵；
num_lower:如果为负，则结果左下为空三角阵。
例：

```
a=tf.constant([[0,1,2,3],[-1,0,1,2],[-2,-1,0,1],[-3,-2,-1,0]])
z=tf.matrix_band_part(a,1,-1)

z==>[[0 1 2 3]
     [-1 0 1 2]
     [0 -1 0 1]
     [0 0 -1 0]]
```

1.59 tf.matrix_set_diag(input,diagonal,name=None)

功能：将输入矩阵的对角元素置换为对角元素。
输入：input：矩阵，diagonal：对角元素。
例：

```
a=tf.constant([[0,1,2,3],[-1,0,1,2],[-2,-1,0,1],[-3,-2,-1,0]])
z=tf.matrix_set_diag(a,[10,11,12,13])

z==>[[10  1  2  3]
     [-1 11  1  2]
     [0  -1 12  1]
     [0   0 -1 13]]
```

1.60 tf.matrix_transpose(a,name='matrix_transpose')

功能：进行矩阵转置。只对低维度的2维矩阵转置，功能同tf.transpose(a,perm=[0,1,3,2])。（若a为4维）
输入：矩阵。

1.61 tf.matmul(a, b, transpose_a=False, transpose_b=False, adjoint_a=False, adjoint_b=False, a_is_sparse=False, b_is_sparse=False, name=None)

功能：矩阵乘法。配置后的矩阵a，b必须满足矩阵乘法对行列的要求。
输入：transpose_a,transpose_b:运算前是否转置；
adjoint_a,adjoint_b:运算前进行共轭；
a_is_sparse,b_is_sparse:a，b是否当作稀疏矩阵进行运算。
例：

```
a = tf.constant([1, 2, 3, 4, 5, 6], shape=[2, 3])
b = tf.constant([7, 8, 9, 10, 11, 12], shape=[3, 2])
z = tf.matmul(a, b)

z==>[[ 58  64]
     [139 154]]
```

1.62 tf.norm(tensor, ord='euclidean', axis=None, keep_dims=False, name=None)

功能：求取范数。
输入：ord：范数类型，默认为‘euclidean’，支持的有‘fro’，‘euclidean’，‘0’，‘1’，‘2’，‘np.inf’；
axis：默认为‘None’，tensor为向量。
keep_dims:默认为‘None’，结果为向量，若为True，保持维度。
例：

```
a = tf.constant([1, 2, 3, 4, 5, 6], shape=[2, 3],dtype=tf.float32)
z = tf.norm(a)
z2=tf.norm(a,ord=1)
```

```
z3=tf.norm(a,ord=2)
z4=tf.norm(a,ord=1,axis=0)
z5=tf.norm(a,ord=1,axis=1)
z6=tf.norm(a,ord=1,axis=1, keep_dims=True)

z==>9.53939
z2==>21.0
z3==>9.53939
z4==>[5.  7.  9.]
z5==>[6.  15.]
z6==>[[6.]
      [15.]]
```

1.63 tf.matrix_determinant(input, name=None)

功能：求行列式。
输入：必须是float32，float64类型。
例：

```
a = tf.constant([1, 2, 3, 4],shape=[2,2],dtype=tf.float32)
z = tf.matrix_determinant(a)

z==>-2.0
```

1.64 tf.matrix_inverse(input, adjoint=None, name=None)

功能：求矩阵的逆。
输入：输入必须是float32，float64类型。adjoint表示计算前求转置
例：

```
a = tf.constant([1, 2, 3, 4],shape=[2,2],dtype=tf.float64)
z = tf.matrix_inverse(a)

z==>[[-2.    1.]
     [1.5  -0.5]]
```

1.65 tf.cholesky(input, name=None)

功能：进行cholesky分解。
输入：注意输入必须是正定矩阵。
例：

```
a = tf.constant([2, -2, -2, 5],shape=[2,2],dtype=tf.float64)
z = tf.cholesky(a)

z==>[[ 1.41421356  0.        ]
     [-1.41421356  1.73205081]]
```

1.66 tf.cholesky_solve(chol, rhs, name=None)

功能：对方程‘AX=RHS’进行cholesky求解。
输入：chol=tf.cholesky(A)。
例：

```
a = tf.constant([2, -2, -2, 5],shape=[2,2],dtype=tf.float64)
chol = tf.cholesky(a)
RHS=tf.constant([3,10],shape=[2,1],dtype=tf.float64)
z=tf.cholesky_solve(chol,RHS)

z==>[[5.83333333]
     [4.33333333]] #A*X=RHS
```

1.67 tf.matrix_solve(matrix, rhs, adjoint=None, name=None)

功能：求线性方程组，matrix*X=rhs。
输入：adjoint:是否对matrix转置。
例：

```
a = tf.constant([2, -2, -2, 5],shape=[2,2],dtype=tf.float64)
RHS=tf.constant([3,10],shape=[2,1],dtype=tf.float64)
z=tf.matrix_solve(a,RHS)

z==>[[5.83333333]
     [4.33333333]]
```

1.68 tf.matrix_triangular_solve(matrix, rhs, lower=None, adjoint=None, name=None)

功能：求解matrix×X=rhs，matrix为上三角或下三角阵。
输入：lower：默认为None，matrix上三角元素为0;若为True，matrix下三角元素为0;
    adjoint：转置
例：

```
a = tf.constant([2, 4, -2, 5],shape=[2,2],dtype=tf.float64)
RHS=tf.constant([3,10],shape=[2,1],dtype=tf.float64)
z=tf.matrix_triangular_solve(a,RHS)
```

```
z==>[[1.5]
     [2.6]]
```

1.69 tf.matrix_solve_ls(matrix, rhs, l2_regularizer=0.0, fast=True, name=None)

功能：求解多个线性方程的最小二乘问题。
输入：。
例：

```
a = tf.constant([2, 4, -2, 5],shape=[2,2],dtype=tf.float64)
RHS=tf.constant([3,10],shape=[2,1],dtype=tf.float64)
z=tf.matrix_solve_ls(a,RHS)

z==>[[-1.38888889]
     [1.44444444]]
```

1.70 tf.qr(input, full_matrices=None, name=None)

功能：对矩阵进行qr分解。
输入：。
例：

```
a = tf.constant([1,2,2,1,0,2,0,1,1],shape=[3,3],dtype=tf.float64)
q,r=tf.qr(a)

q==>[[-0.70710678   0.57735027   -0.40824829]
     [-0.70710678  -0.57735027    0.40824829]
     [0.            0.57735027   0.81649658 ]]
r==>[[-1.41421356  -1.41421356   -2.82842712]
     [0.            1.73205081    0.57735027]
     [0.            0.            0.81649658]]
```

1.71 tf.self_adjoint_eig(tensor, name=None)

功能：求取特征值和特征向量。
输入：
例：

```
a = tf.constant([3,-1,-1,3],shape=[2,2],dtype=tf.float64)
e,v=tf.self_adjoint_eig(a)

e==>[2.  4.]
v==>[[0.70710678 0.70710678]
    [0.70710678 -0.70710678]]
```

1.72 tf.self_adjoint_eigvals(tensor, name=None)

功能：计算多个矩阵的特征值。
输入：shape=[....,N,N]。

1.73 tf.svd(tensor, full_matrices=False, compute_uv=True, name=None)

功能：进行奇异值分解。tensor=u×diag（s）×transpose（v）
输入：
例：

```
a = tf.constant([3,-1,-1,3],shape=[2,2],dtype=tf.float64)
s,u,v=tf.svd(a)

s==>[4. 2.]
u==>[[0.70710678 0.70710678]
    [-0.70710678 0.70710678]]
v==>[[0.70710678 0.70710678]
    [-0.70710678 0.70710678]]
```

1.74 tf.tensordot(a, b, axes, name=None)

功能：同numpy.tensordot，根据axis计算点乘。
输入：axes=1或axes=[[1],[0]]，即为矩阵乘。
例：

```
a = tf.constant([1,2,3,4],shape=[2,2],dtype=tf.float64)
b = tf.constant([1,2,3,4],shape=[2,2],dtype=tf.float64)
z=tf.tensordot(a,b,axes=[[1],[1]])

z==>[[5.  11.]
     [11. 25.]]
```

1.75 tf.complex(real, imag, name=None)

功能：将实数转化为复数。
输入：real，imag：float32或float64。
例：

```
real = tf.constant([1,2],dtype=tf.float64)
imag = tf.constant([3,4],dtype=tf.float64)
z=tf.complex(real,imag)
```

```
z==>[1.+3.j  2.+4.j]
```

1.76 tf.conj(x, name=None)

功能：返回x的共轭复数。
输入：
例：

```
a = tf.constant([1+2j,2-3j])
z=tf.conj(a)

z==>[1.-2.j  2.+3.j]
```

1.77 tf.imag(input, name=None)

功能：返回虚数部分。
输入：`complex64`,`complex128`类型。
例：

```
a = tf.constant([1+2j,2-3j])
z=tf.imag(a)

z==>[2.  -3.]
```

1.78 tf.real(input,name=None)

功能：返回实数部分。
输入：`complex64`,`complex128`类型。
例：

```
a = tf.constant([1+2j,2-3j])
z=tf.real(a)

z==>[1.  2.]
```

1.79 ～1.84 fft变换

函数：

```
tf.fft(input, name=None)
tf.ifft(input, name=None)
tf.fft2d(input, name=None)
tf.ifft2d(input, name=None)
tf.fft3d(input, name=None)
tf.ifft3d(input, name=None)
```