Prof. Dr. L. Paditz, 12.02.2019

HTW Dresden,

paditz@htw-dresden.de

Mathematik-Grundlagen als eActivty mit ClassPad

Gemischte Aufgaben in TensorFlow

==========

```
parallels@parallels-Parallels-Virtual-Platform:~$ python3
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for
more information.
>>> import tensorflow as tf
>>> '''Einfache mathematische Operationen mit
TensorFlow'''
'Einfache mathematische Operationen mit TensorFlow'
>>>
```

```
>>> # Berechnungen mit konstanten Tensoren
...
>>> const_a = tf.constant(3.6)
# Tensor der Dimension 0 (rank=0)
>>> const_a  # Tensor der Dim. 0, Tensor mit nur
einem Wert
<tf.Tensor 'Const:0' shape=() dtype=float32>
>>> const_b = tf.constant(1.2)
>>> total = const_a + const_b
>>> quot = tf.div(const_a,const_b)
>>> tensor1 = [const_a,const_b,total,quot]
>>> sess = tf.Session()
>>> print(sess.run(tensor1))
[3.6, 1.2, 4.8, 2.9999998]
>>> print("Quotient: %f" % sess.run(quot))
Quotient: 3.000000
>>> print("%f" % sess.run(quot))
3.000000
```

```
============================

String Formatting in Python using %

%d - integer

%f - float

%s - string

%x - hexadecimal

%o - octal

============================


const_a:=3.6
                                                        3.6
const_b:=1.2
                                                        1.2
total:=const_a+const_b
                                                        4.8
quot:=const_a/const_b
                                                          3
stop
============================
>>>
>>> # Berechnungen mit zufälligen Tensoren
...
```

```
>>> rand_a = tf.random_normal([5],2.0) # Vorgabe
μ=2., Standardvorgabe σ=1.
>>> print(sess.run(rand_a))
[2.2870543 1.4474564 2.7151756 1.3589697
1.6633828]
>>> rand_b = tf.random_uniform([5],1.0,4.0) #
Vorgabe min=1., max=4.
>>> diff = rand_a - rand_b
>>> print(sess.run(diff))
[-0.64230037 -2.0911734   2.4180498
-1.5646839  -0.740116  ]
>>> diff = tf.subtract(rand_a,rand_b)
>>> print(sess.run(diff))
[-0.66464055 -1.1234436  -3.1447835
-1.1904049  -0.5177622 ]
>>> # Zufallszahlen werden erst mit dem
sess.run-Befehl generiert!
>>>
>>> print(sess.run(rand_a))
[0.4336846 1.7447778 2.1962774 2.1241724
4.0425243]
>>> print(sess.run(rand_a))
[1.167691  2.837778  1.2518945 1.7713184
0.5472362]
==========
```

randNorm(σ,μ,n) Zufallszahlen als Liste {...}

```
rand_a:=randNorm(9,5,50)
     {6.085589646,12.57886101,6.18351574,-10.9827▶
```

mean(rand_a)

4.01621882

variance(rand_a)

60.70844418

stdDev(rand_a)

7.791562371

randList(n) Zufallszahlen(gleichverteilt) aus (0,1) als Liste {...}

Zufallszahlen aus (1,4)

rand_b:=1+3*randList(50)

{3.582678928,3.225831978,3.473163173,2.16270▶

mean(rand_b)

2.620162127

diff() ist bei ClassPad eine Systemfunktion!

diff ist kein zulässiger freier Variablenname!

daher **diffab** statt **diff** gewählt:

Listenarithmetik:

diffab:=rand_a-rand_b

{2.502910718,9.353029029,2.710352567,-13.145▶

stop

==========

>>>

>>> **# Vektormultiplikation (Tensormultiplikation)**

...

>>> vec_a = **tf.linspace**(0.,3.,4) # Wertsequenz von 0. bis 3. mit 4 Werten generieren

>>> vec_b = **tf.fill**([4,1],2.) # Tensor mit Shape[4,1] mit den Werten 2. füllen

>>> print(sess.run(vec_a))

```
[0. 1. 2. 3.]
>>> print(sess.run(vec_b))
[[2.]
 [2.]
 [2.]
 [2.]]
============================
vec_a:=[0.,1.,2.,3.]
```

$$[0\ 1\ 2\ 3]$$

```
vec_b:=fill(2,4,1)
```

$$\begin{bmatrix}2\\2\\2\\2\end{bmatrix}$$

```
vec_b*vec_a
```

$$\begin{bmatrix}0&2&4&6\\0&2&4&6\\0&2&4&6\\0&2&4&6\end{bmatrix}$$

Skalarprodukt als Matrix(Tensor) vom Typ(1,1):

```
vec_a*vec_b
```

$$[12]$$

Skalarprodukt als Zahl:

```
dotP(vec_a,trn(vec_b))
```

$$12$$

```
stop
============================
>>> prod = tf.multiply(vec_a,vec_b)
>>> print(sess.run(prod))
[[0. 2. 4. 6.]
```

```
 [0. 2. 4. 6.]
 [0. 2. 4. 6.]
 [0. 2. 4. 6.]]
>>> dot = tf.tensordot(vec_a,vec_b,1)
>>> print(sess.run(dot))
[12.]
>>> prod1 = tf.multiply(vec_b,vec_a)
>>> print(sess.run(prod1))
[[0. 2. 4. 6.]
 [0. 2. 4. 6.]
 [0. 2. 4. 6.]
 [0. 2. 4. 6.]]
====================================
# Die Tensormultiplikation von Tensoren mit
shape=(n,) oder shape=(n, 1) ist kommutativ,
die Vektormultiplikation(Matrixmultiplikation) nicht!
====================================
>>> vec_a  # Zeilenvektor (eindim. Tensor, rank=1)
<tf.Tensor 'LinSpace:0' shape=(4,) dtype=float32>
>>> vec_b  # Spaltenvektor (zweidim. Tensor,
rank=2)
<tf.Tensor 'Fill:0' shape=(4, 1) dtype=float32>
>>> prod   # Matrix (zweidim. Tensor, rank=2)
<tf.Tensor 'Mul:0' shape=(4, 4) dtype=float32>
>>> prod1
<tf.Tensor 'Mul_1:0' shape=(4, 4) dtype=float32>
>>> dot    # Zeilenvektor (eindim. Tensor, rank=1)
<tf.Tensor 'Tensordot_3:0' shape=(1,) dtype=float32>
>>>
```

```
>>> # Matrixmultiplikation
...
>>> mat_a = tf.constant([[2,3],[1,2],[4,5]])
>>> mat_b = tf.constant([[6,4,1],[3,7,2]])
>>> mat_prod = tf.matmul(mat_a,mat_b)
>>> print(sess.run(mat_a))
[[2 3]
 [1 2]
 [4 5]]
>>> print(sess.run(mat_b))
[[6 4 1]
 [3 7 2]]
>>> print(sess.run(mat_prod))
[[21 29  8]
 [12 18  5]
 [39 51 14]]
>>> mat_a  # Tensor mit 3 Elementen, jeweils 2
Spalten
<tf.Tensor 'Const_2:0' shape=(3, 2) dtype=int32>
>>> mat_b  # Tensor mit 2 Elementen, jeweils 3
Spalten
<tf.Tensor 'Const_3:0' shape=(2, 3) dtype=int32>
>>> mat_prod # Tensor mit 3 Elementen, jeweils 3
Spalten
<tf.Tensor 'MatMul:0' shape=(3, 3) dtype=int32>
>>>
==========================
mat_a:=[[2,3],[1,2],[4,5]]
```

$$\begin{bmatrix} 2 & 3 \\ 1 & 2 \\ 4 & 5 \end{bmatrix}$$

```
mat_b:=[[6,4,1],[3,7,2]]
```

$$\begin{bmatrix} 6 & 4 & 1 \\ 3 & 7 & 2 \end{bmatrix}$$

```
mat_prod:=mat_a*mat_b
```

$$\begin{bmatrix} 21 & 29 & 8 \\ 12 & 18 & 5 \\ 39 & 51 & 14 \end{bmatrix}$$

==========

Quelle:

TensorFlow für dummies (2018),
Kapitel 3: Tensoren und Operationen
ISBN 978-3-527-7157-3

Download für dieses Dokument:
www.informatik.htw-dresden.de/
~paditz/Tensorflow-Ue04.pdf