Prof. Dr. L. Paditz, 11.02.2019

HTW Dresden,

paditz@htw-dresden.de

Mathematik-Grundlagen als eActivty mit ClassPad

später dann mit **TensorFlow**, erstes Tensorflow-Beispiel

Informatik ist ein Kunstwort der deutschen Sprache, in dem die Worte **Infor**mation und Mathe**matik** stecken. Die Mathematik ist eine wesentliche Wurzel der Informatik: In allen ihren Bereichen werden immer wieder mathematische Methoden verwendet, **mathematische Vorgehensweisen** sind typisch für die Arbeit des Informatikers.

Matrizenrechnung:

Führen Sie die folgende Matrixmultiplikation durch:

a) traditionelle Rechnung

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

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

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

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

C:=A·B

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

anderes Eingabeformat:

Zeilen durch Kommata getrennt,

außen und innen: **eckige** Klammern nutzen

```
mat_a:=[[2,3],[1,2],[4,5]]
```

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

Zeilen müssen nicht notwendig durch Kommata getrennt sein, jedoch bei **Tensorflow** sind die Kommata als Trennzeichen erforderlich:

```
[[2,3][1,2][4,5]]
```

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

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

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

[[6,4,1][3,7,2]]

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

mat_prod:=mat_a·mat_b

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

Tensorflow:

==========

Skript

```
python3
import tensorflow as tf
# 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)
# Ausführung der Operationen
with tf.Session() as sess:
    print("Matrixprodukt: ", sess.run(mat_prod))
```

==========

Rechnerprotokoll:

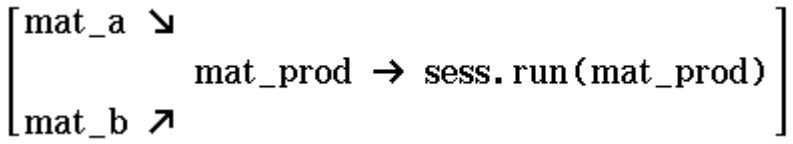
```
parallels@parallels-Parallels-Virtual-Platform:~$ python3

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.

>>> '''Einfache mathematische Operationen mit
TensorFlow'''
'Einfache mathematische Operationen mit TensorFlow'

>>> # Die folgenden 3 Zeilen sind nur für Python2
erforderlich!
>>> from __future__ import absolute_import
>>> from __future__ import division
>>> from __future__ import print_function

>>> import tensorflow as tf

>>> # 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)

>>> # Ausführung der Operationen
>>> with tf.Session() as sess:
...       print("Matrixprodukt: ",
sess.run(mat_prod))
Matrixprodukt: [[21 29  8]
                [12 18  5]
                [39 51 14]]
```

>>>

Unterschied zum traditionellem Rechnen:
Zuerst werden die Eingaben und Rechenoperationen definiert.
Die Abarbeitung der Befehle startet erst nach einem Aufruf von **sess.run().**

Ablauf-Graph: Datenfluss-Modell, TensorFlow-Graph
Operationen als Knoten



TensorFlow nutzt das **datenstrom-orientierte** Paradigma. In diesem wird ein **Datenfluss-Berechnungsgraph** erstellt, welcher aus **Knoten** und **Kanten** besteht. Ein Datenfluss-Berechnungsgraph, **Datenflussgraph** oder auch **Berechnungsgraph** kann mehrere Knoten haben, die wiederum durch die Kanten verbunden sind. In TensorFlow steht jeder **Knoten für eine Operation,** die Auswirkungen auf eingehende Daten haben.

Beispiel: y=**W**·x+b

$$W := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$x := \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$b := \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$$

$$\begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$$

y:=W*x+b

$$\begin{bmatrix} 18 \\ 37 \\ 56 \end{bmatrix}$$

Zwischenergebnis:

W•x

$$\begin{bmatrix} 14 \\ 32 \\ 50 \end{bmatrix}$$

Tensorflow: Skript

```
python3
import tensorflow as tf
W = tf.constant([[1,2,3],[4,5,6],[7,8,9]])
print(W)
x = tf.constant([[1],[2],[3]])
```

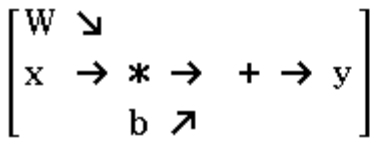
```
print(x)
b = tf.constant([[4],[5],[6]])
print(b)
Wx = tf.matmul(W,x)
y = tf.add(Wx,b)
with tf.Session() as sess:
    print(y)
    print("Zwischenergebnis: ", sess.run(Wx))
    print("Endergebnis: ", sess.run(y))
```

Rechnerprotokoll:

```
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
>>> W = tf.constant([[1,2,3],[4,5,6],[7,8,9]])
>>> print(W)
Tensor("Const:0", shape=(3, 3), dtype=int32)
>>> x = tf.constant([[1],[2],[3]])
>>> print(x)
Tensor("Const_1:0", shape=(3, 1), dtype=int32)
>>> b = tf.constant([[4],[5],[6]])
>>> print(b)
Tensor("Const_2:0", shape=(3, 1), dtype=int32)
>>> Wx = tf.matmul(W,x)
>>> y = tf.add(Wx,b)
>>> with tf.Session() as sess:
```

```
...         print(y)
...         print("Zwischenergebnis: ", sess.run(Wx))
...         print("Endergebnis: ", sess.run(y))
...
Tensor("Add:0", shape=(3, 1), dtype=int32)
Zwischenergebnis:  [[14]
                    [32]
                    [50]]
Endergebnis:  [[18]
               [37]
               [56]]
>>>
```

Ablauf-Graph: Datenfluss-Modell



Ein Tensor weist drei Eigenschaften auf: **Rang**, **Form**(Shape) und **Typ**. Ein Rang entspricht der Anzahl an Dimensionen eines Tensors.

W: Rank: 2, entspricht der Anzahl der notwendigen Koordinaten eines Tensors (= Dimension des Tensors)
Shape: (3,3) entspricht der Dimension {3,3} der Matrix:
dim(W)

{3,3}

y: Rank: 2, Der Spaltenvektor ist hier eine 2-dim.

Matrix
Shape: (3,1) entspricht der Dimension {3,1} der Matrix:
dim(y)

{3,1}

Unter einem **Typ** versteht man den **Datentyp** in jeder Koordinate des Tensors.

Ist ein TensorFlow-Graph schließlich erstellt, so kann noch nicht sofort z.B. mit der Bilderkennung etc. begonnen werden. Für diese Aufgabe muss er erst vorbereitet bzw. "trainiert" werden. Das heißt, das System muss dazu iterativ mit Trainingsdaten gefüttert werden. Gleichzeitig werden die Gewichte innerhalb des Graphen so verändert, dass der Output sich dem erwarteten Ausgabewert annähert (approximiert). Bei dieser Vorgehensweise spielt die **Wahrscheinlichkeitsrechnung** eine große Rolle und zählt zu den ganz entscheidenden technologischen Durchbrüchen der vergangenen Jahre.

Das kleine Programm **hello_tensorflow.py** kann in der Kommandozeile mit **"python3 hello_tensorflow.py"** gestartet werden. Wenn jetzt "Hallo World" richtig angezeigt wird, ist **TensorFlow** korrekt installiert:

```
========================================
python3
''' Ein einfaches TensorFlow-Programm '''
```

```
# Die folgenden 3 Zeilen sind nur für Python2
erforderlich!
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import tensorflow as tf

# Tensor erzeugen
msg = tf.string_join(['Hallo ', 'TensorFlow!'])

# Sitzung starten
with tf.Session() as sess:
    print(sess.run(msg))
=======================================
```

kürzer: andere Syntax mit **tf.Session()**
zuerst Terminal starten:

```
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
>>> hello = tf.constant('Hello World')
>>> sess = tf.Session()
>>> print(sess.run(hello))
b'Hello World'
```

>>>

z.B. Tensor der Dimension {3,3,3}, d.h.
Shape(3,3,3) und Rank=3:

```
[[[1,2,3],[4,5,6],[7,8,9]],
 [[10,11,12],[13,14,15],[16,17,18]],
 [[19,20,21],[22,23,24],[25,26,27]]]
```

bedeutet:

[[1,2,3],[4,5,6],[7,8,9]]

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

[[10,11,12],[13,14,15],[16,17,18]]

$$\begin{bmatrix} 10 & 11 & 12 \\ 13 & 14 & 15 \\ 16 & 17 & 18 \end{bmatrix}$$

[[19,20,21],[22,23,24],[25,26,27]]

$$\begin{bmatrix} 19 & 20 & 21 \\ 22 & 23 & 24 \\ 25 & 26 & 27 \end{bmatrix}$$

$$\left[\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \begin{bmatrix} 10 & 11 & 12 \\ 13 & 14 & 15 \\ 16 & 17 & 18 \end{bmatrix}, \begin{bmatrix} 19 & 20 & 21 \\ 22 & 23 & 24 \\ 25 & 26 & 27 \end{bmatrix}\right]$$, d.h.

der Tensor besteht aus 3 Matrizen, ist also eine Blockmatrix!

Download für dieses Dokument:

www.informatik.htw-dresden.de/
~paditz/Tensorflow-Ue02.pdf