Prof. Dr. L. Paditz, 15.02.2019

HTW Dresden,

paditz@htw-dresden.de

Mathematik-Grundlagen als eActivty mit ClassPad

multivariate lineare Regression in TensorFlow

Gradientenverfahren - steilster Abstieg zum Min!

Quelle:

Einführung in Tensorflow

ISBN 978-3-96009-074-8

Beispiel 1: **multivariate lineare Regression**, S. 44ff

Modellgleichung:

mit $[w_1, w_2, w_3]=[0.3, 0.5, 0.1]$, $b=-0.2$, $i=1(1)n$,

für die Simulation verrauschter Daten.

Define $y(w_1, w_2, w_3, b, x_{1i}, x_{2i}, x_{3i}) = \text{dotP}([w_1, w_2, w_3], [x$ ▶

done

$y(w_1, w_2, w_3, b, x_{1i}, x_{2i}, x_{3i})$

$$w_1 \cdot x_{1i} + w_2 \cdot x_{2i} + w_3 \cdot x_{3i} + b$$

$$\text{loss}(w_1, w_2, w_3, b) = \frac{1}{n} \sum_{i=1}^{n} \left((y(w_1, w_2, w_3, b, x_{1i}, x_{2i}, x_{3i}) - y_i \right.$$ ▶

Define $\text{loss}_i(w_1, w_2, w_3, b) = (y(w_1, w_2, w_3, b, x_{1i}, x_{2i}, x_{3i}) -$ ▶

done

$\text{loss}_i(w_1, w_2, w_3, b)$

$$(w_1 \cdot x_{1i} + w_2 \cdot x_{2i} + w_3 \cdot x_{3i} + b - y_i)^2$$

Gradient:

$$\begin{bmatrix} \frac{d}{dw_1}(\text{loss}_i(w_1, w_2, w_3, b)) \\ \frac{d}{dw_2}(\text{loss}_i(w_1, w_2, w_3, b)) \\ \frac{d}{dw_3}(\text{loss}_i(w_1, w_2, w_3, b)) \\ \frac{d}{db}(\text{loss}_i(w_1, w_2, w_3, b)) \end{bmatrix}$$

$$\begin{bmatrix} 2 \cdot x_{1i} \cdot (w_1 \cdot x_{1i} + w_2 \cdot x_{2i} + w_3 \cdot x_{3i} + b - y_i) \\ 2 \cdot x_{2i} \cdot (w_1 \cdot x_{1i} + w_2 \cdot x_{2i} + w_3 \cdot x_{3i} + b - y_i) \\ 2 \cdot x_{3i} \cdot (w_1 \cdot x_{1i} + w_2 \cdot x_{2i} + w_3 \cdot x_{3i} + b - y_i) \\ 2 \cdot (w_1 \cdot x_{1i} + w_2 \cdot x_{2i} + w_3 \cdot x_{3i} + b - y_i) \end{bmatrix}$$

$\text{expand}(\text{ans}/2)$

$$\begin{bmatrix} w_1 \cdot x_{1i}^2 + w_2 \cdot x_{1i} \cdot x_{2i} + w_3 \cdot x_{1i} \cdot x_{3i} + b \cdot x_{1i} - x_{1i} \cdot y_i \\ w_2 \cdot x_{2i}^2 + w_1 \cdot x_{1i} \cdot x_{2i} + w_3 \cdot x_{2i} \cdot x_{3i} + b \cdot x_{2i} - x_{2i} \cdot y_i \\ w_3 \cdot x_{3i}^2 + w_1 \cdot x_{1i} \cdot x_{3i} + w_2 \cdot x_{2i} \cdot x_{3i} + b \cdot x_{3i} - x_{3i} \cdot y_i \\ w_1 \cdot x_{1i} + w_2 \cdot x_{2i} + w_3 \cdot x_{3i} + b - y_i \end{bmatrix}$$

analytische Lösung nach MKQ:

$\text{grad}(\text{loss}) =$

$$\frac{2}{n}\cdot\begin{bmatrix}\sum_{i=1}^{n}\left(w_1\cdot x_{1i}^2+w_2\cdot x_{1i}\cdot x_{2i}+w_3\cdot x_{1i}\cdot x_{3i}+b\cdot x_{1i}-x_{1i}\cdot y_i\right)\\ \sum_{i=1}^{n}\left(w_2\cdot x_{2i}^2+w_1\cdot x_{1i}\cdot x_{2i}+w_3\cdot x_{2i}\cdot x_{3i}+b\cdot x_{2i}-x_{2i}\cdot y_i\right)\\ \sum_{i=1}^{n}\left(w_3\cdot x_{3i}^2+w_1\cdot x_{1i}\cdot x_{3i}+w_2\cdot x_{2i}\cdot x_{3i}+b\cdot x_{3i}-x_{3i}\cdot y_i\right)\\ \sum_{i=1}^{n}\left(w_1\cdot x_{1i}+w_2\cdot x_{2i}+w_3\cdot x_{3i}+b-y_i\right)\end{bmatrix}$$

$$=\begin{bmatrix}0\\0\\0\\0\end{bmatrix}$$

$$\begin{bmatrix}\sum_{i=1}^{n}\left(x_{1i}^2\right) & \sum_{i=1}^{n}\left(x_{1i}\cdot x_{2i}\right) & \sum_{i=1}^{n}\left(x_{1i}\cdot x_{3i}\right) & \sum_{i=1}^{n}\left(x_{1i}\right)\\ \sum_{i=1}^{n}\left(x_{1i}\cdot x_{2i}\right) & \sum_{i=1}^{n}\left(x_{2i}^2\right) & \sum_{i=1}^{n}\left(x_{2i}\cdot x_{3i}\right) & \sum_{i=1}^{n}\left(x_{2i}\right)\\ \sum_{i=1}^{n}\left(x_{1i}\cdot x_{3i}\right) & \sum_{i=1}^{n}\left(x_{2i}\cdot x_{3i}\right) & \sum_{i=1}^{n}\left(x_{3i}^2\right) & \sum_{i=1}^{n}\left(x_{3i}\right)\\ \sum_{i=1}^{n}\left(x_{1i}\right) & \sum_{i=1}^{n}\left(x_{2i}\right) & \sum_{i=1}^{n}\left(x_{3i}\right) & n\end{bmatrix}$$

$$\cdot\begin{bmatrix}w_1\\w_2\\w_3\\b\end{bmatrix}=\begin{bmatrix}\sum_{i=1}^{n}\left(x_{1i}\cdot y_i\right)\\ \sum_{i=1}^{n}\left(x_{2i}\cdot y_i\right)\\ \sum_{i=1}^{n}\left(x_{3i}\cdot y_i\right)\\ \sum_{i=1}^{n}\left(y_i\right)\end{bmatrix}$$

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ b \end{bmatrix} =$$

$$\begin{bmatrix} \sum_{i=1}^{n}(x_{1i}^2) & \sum_{i=1}^{n}(x_{1i}\cdot x_{2i}) & \sum_{i=1}^{n}(x_{1i}\cdot x_{3i}) & \sum_{i=1}^{n}(x_{1i}) \\ \sum_{i=1}^{n}(x_{1i}\cdot x_{2i}) & \sum_{i=1}^{n}(x_{2i}^2) & \sum_{i=1}^{n}(x_{2i}\cdot x_{3i}) & \sum_{i=1}^{n}(x_{2i}) \\ \sum_{i=1}^{n}(x_{1i}\cdot x_{3i}) & \sum_{i=1}^{n}(x_{2i}\cdot x_{3i}) & \sum_{i=1}^{n}(x_{3i}^2) & \sum_{i=1}^{n}(x_{3i}) \\ \sum_{i=1}^{n}(x_{1i}) & \sum_{i=1}^{n}(x_{2i}) & \sum_{i=1}^{n}(x_{3i}) & n \end{bmatrix}^{-1}$$

$$\cdot \begin{bmatrix} \sum_{i=1}^{n}(x_{1i}\cdot y_i) \\ \sum_{i=1}^{n}(x_{2i}\cdot y_i) \\ \sum_{i=1}^{n}(x_{3i}\cdot y_i) \\ \sum_{i=1}^{n}(y_i) \end{bmatrix}$$

mit tensorflow Datensatz: 2000 Vektoren $[x_{1i}, x_{2i}, x_{3i}]$, i=1(1)2000:

```
>>> print(x_data)
[[-0.58111234  0.83820954  1.5950554 ]
 [-0.0136423  -1.27946206  0.10544973]
 [-0.22532237  1.35862731 -0.9139968 ]
 ...
```

```
 [ 0.24275082 -0.76875081  0.10545472]
 [-0.65309978  0.38736633  0.73586307]
 [ 1.96254475  0.25914792 -0.09957582]]
```

Modell: $y_i=w_1 \cdot x_{1i}+w_2 \cdot x_{2i}+w_3 \cdot x_{3i}+b$ mit unbekannten w_1, w_2, w_3, b

2000 beobachtete y_i, i=1(1)2000, **mit gaußschem Rauschen:**

$y_i=w_1 \cdot x_{1i}+w_2 \cdot x_{2i}+w_3 \cdot x_{3i}+b+noise_i$

```
>>> print(y_data)
[[ 0.16173649 -0.72014297  0.31910246 ...
-0.62691472  0.01518871
   0.46835219]]
```

für die geg. x_data seien nur die verrauschten y_data bekannt,

ges. sind die Modellparameter w_1, w_2, w_3, b

Simulation: mit n=10:

DelVar x, y, w_1, w_2, w_3, b

done

Define $y(x_{1i}, x_{2i}, x_{3i})=dotP([0.3,0.5,0.1],[x_{1i}, x_{2i}, x$ ▶

done

$y(x_{1i}, x_{2i}, x_{3i})$

$$0.3 \cdot x_{1i}+0.5 \cdot x_{2i}+0.1 \cdot x_{3i}-0.2$$

$listx_1:=randNorm(1,0,10)$

{-0.8470227791,-1.805372355,1.437810785,0.35 ▶

$listx_2:=randNorm(1,0,10)$

{-1.076641538,0.7035196326,0.7032136215,-1.0 ▶

$listx_3:=randNorm(1,0,10)$

{0.1653038676, 0.9652507237, 0.5289016885, −0.9 ▶

noise:=randNorm(1, 0, 10)·0.1

{−0.05317682153, −0.1161420792, 0.08695041832, ▶

x_1:=listToMat($listx_1$)

$$\begin{bmatrix} -0.8470227791 \\ -1.805372355 \\ 1.437810785 \\ 0.3504267099 \\ 1.266706869 \\ 0.2767806922 \\ 1.223438761 \\ -0.6929037841 \\ -2.309888374 \\ 0.5938545878 \end{bmatrix}$$

x_2:=listToMat($listx_2$)

$$\begin{bmatrix} -1.076641538 \\ 0.7035196326 \\ 0.7032136215 \\ -1.007421554 \\ -0.1444805193 \\ -1.235365515 \\ 2.195655187 \\ -0.07135534161 \\ -0.09994577534 \\ -0.07777053588 \end{bmatrix}$$

x_3:=listToMat($listx_3$)

$$\begin{bmatrix} 0.1653038676 \\ 0.9652507237 \\ 0.5289016885 \\ -0.9701846283 \\ 0.1842763127 \\ -1.209833512 \\ -1.51183059 \\ -0.1467117226 \\ -0.507764299 \\ 0.6159099134 \end{bmatrix}$$

noi:=listToMat(noise)

$$\begin{bmatrix} -0.05317682153 \\ -0.1161420792 \\ 0.08695041832 \\ -0.06815418022 \\ -0.09567509987 \\ -0.08419417007 \\ 0.1205548881 \\ 1.687043608\text{E}{-3} \\ 0.02663015176 \\ -0.09548008865 \end{bmatrix}$$

$$y_{noi}:=0.3\cdot x_1+0.5\cdot x_2+0.1\cdot x_3-0.2\cdot \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} +noi$$

$$\begin{bmatrix} -1.029074038 \\ -0.4094688971 \\ 0.7227906335 \\ -0.763755407 \\ 0.03052433244 \\ -0.9398260709 \\ 1.234231051 \\ -0.4565329347 \\ -0.967085678 \\ -0.09461798893 \end{bmatrix}$$

list1:=matToList(ans, 1)

{-1.029074038,-0.4094688971,0.7227906335,-0. ▶

DelVar y

done

$$y:=0.3\cdot x_1+0.5\cdot x_2+0.1\cdot x_3-0.2\cdot \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} -0.9758972161 \\ -0.2933268179 \\ 0.6358402152 \\ -0.6956012268 \\ 0.1261994323 \\ -0.8556319008 \\ 1.113676162 \\ -0.4582199783 \\ -0.9937158298 \\ 8.620997209E-4 \end{bmatrix}$$

list2:=matToList(ans, 1)

{−0.9758972161, −0.2933268179, 0.6358402152, −0 ▶

STAT−Menü 

stop

Simulation:

Erhöhung auf n=200:

n:=200

200

$listx_1$:=randNorm(1, 0, n)

{−0.5029788358, 0.4449061287, −1.445662176, 1.2 ▶

$listx_2$:=randNorm(1, 0, n)

{−0.9384077334, −0.2381210971, 0.120616003, 1.8 ▶

$listx_3$:=randNorm(1, 0, n)

{1.019956287, −1.621866608, 1.03731305, 1.33465 ▶

noise:=randNorm(1, 0, n)·0.1

{0.1874571211, 8.20411663E-3, 0.04287236322, −0. ▶

list3:=0.3·$listx_1$+0.5·$listx_2$+0.1·$listx_3$−0.2+noise

{−0.5306447676, −0.3395712541, −0.4267869831, 1 ▶

list4:=0.3·$listx_1$+0.5·$listx_2$+0.1·$listx_3$−0.2

{−0.7181018888, −0.3477753707, −0.4696593463, 1 ▶

STAT−Menü 

Punkteplot mit künstlich erzeugten Daten

MKQ−Schätzung:

$$\begin{bmatrix} \text{sum}(listx_1 * list3) \\ \text{sum}(listx_2 * list3) \\ \text{sum}(listx_3 * list3) \\ \text{sum}(list3) \end{bmatrix}$$

$$\begin{bmatrix} 57.24885662 \\ 94.19583837 \\ 21.13464718 \\ -40.43049868 \end{bmatrix}$$

$$\begin{bmatrix} \text{sum}(\text{listx}_1{}^2) & \text{sum}(\text{listx}_1 \cdot \text{listx}_2) & \text{sum}(\text{listx}_1 \cdot \text{listx}_3) & \text{s} \\ \text{sum}(\text{listx}_1 \cdot \text{listx}_2) & \text{sum}(\text{listx}_2{}^2) & \text{sum}(\text{listx}_2 \cdot \text{listx}_3) & \text{s} \\ \text{sum}(\text{listx}_1 \cdot \text{listx}_3) & \text{sum}(\text{listx}_2 \cdot \text{listx}_3) & \text{sum}(\text{listx}_3{}^2) & \text{s} \\ \text{sum}(\text{listx}_1) & \text{sum}(\text{listx}_2) & \text{sum}(\text{listx}_3) & \text{n} \end{bmatrix}$$

$$\begin{bmatrix} 165.9324114 & 10.27443745 & -3.094904615 & -13.2 \\ 10.27443745 & 186.6102139 & 10.12384103 & 6.542 \\ -3.094904615 & 10.12384103 & 167.0549894 & -1.23 \\ -13.22011757 & 6.542625009 & -1.230482855 & 200 \end{bmatrix}$$

$$\text{ans}^{-1} \cdot \begin{bmatrix} 57.24885662 \\ 94.19583837 \\ 21.13464718 \\ -40.43049868 \end{bmatrix}$$

$$\begin{bmatrix} 0.30082813 \\ 0.4896633602 \\ 0.100955885 \\ -0.1976648735 \end{bmatrix}$$

Parameterschätzung: $\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ b \end{bmatrix} = \begin{bmatrix} 0.30082813 \\ 0.4896633602 \\ 0.100955885 \\ -0.1976648735 \end{bmatrix} \approx \begin{bmatrix} 0.3 \\ 0.5 \\ 0.1 \\ -0.2 \end{bmatrix}$

Skript:

======

python3

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL']='2'
```

```
import time
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
# === Erzeuge Daten und simuliere Ergebnisse ====

x_data = np.random.randn(2000,3)
w_real = [0.3,0.5,0.1]
b_real = -0.2
noise = np.random.randn(1,2000)*0.1
y_data = np.matmul(w_real,x_data.T) + b_real +
noise

NUM_STEPS = 10
g = tf.Graph()
wb_ = []
with g.as_default():
    x = tf.placeholder(tf.float32,shape=[None,3])
    y_true = tf.placeholder(tf.float32,shape=None)
    with tf.name_scope('inference') as scope:
        w =
tf.Variable([[0,0,0]],dtype=tf.float32,name='weights')
        b = tf.Variable(0,dtype=tf.float32,name='bias')
        y_pred = tf.matmul(w,tf.transpose(x)) + b
    with tf.name_scope('loss') as scope:
        loss =
tf.reduce_mean(tf.square(y_true-y_pred))
    with tf.name_scope('train') as scope:
        learning_rate = 0.5
```

```
        optimizer =
tf.train.GradientDescentOptimizer(learning_rate)
        train = optimizer.minimize(loss)
    # Initialisierung der Variablen vor Beginn. Wird als
Erstes ausgeführt.
    init = tf.global_variables_initializer()
    with tf.Session() as sess:
        sess.run(init)
        for step in range(NUM_STEPS):
            sess.run(train,{x: x_data, y_true: y_data})
            if step % 5 == 0:
                print(step, sess.run([w,b]))
                wb_.append(sess.run([w,b]))
        print(10, sess.run([w,b]))

print(x_data)
print(y_data)
print(noise)

# plot the results
realy_data = np.matmul(w_real,x_data.T) + b_real
predictedy_data = np.matmul(w_real,x_data.T) +
b_real + noise
plt.plot(realy_data, predictedy_data, 'bo',
label='Predicted y_data')
plt.plot(realy_data, realy_data, 'ro', label='Real
y_data')
plt.legend()
plt.show()
```

Rechnerdurchlauf:

```
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 numpy as np
>>> import tensorflow as tf
>>> # === Erzeuge Daten und simuliere Ergebnisse
====
...
>>> x_data = np.random.randn(2000,3)
>>> w_real=[[0.3,0.5,0.1]]
>>> b_real=-0.2
>>> noise = np.random.randn(1,2000)*0.1
>>> y_data = np.matmul(w_real,x_data.T) + +b_real
+ noise
>>>
>>> NUM_STEPS = 10
>>> g = tf.Graph()
>>> wb_ = []
>>> with g.as_default():
...     x = tf.placeholder(tf.float32,shape=[None,3])
...     y_true = tf.placeholder(tf.float32,shape=None)
...     with tf.name_scope('inference') as scope:
...         w =
```

```
tf.Variable([[0,0,0]],dtype=tf.float32,name='weights')
...         b =
tf.Variable(0,dtype=tf.float32,name='bias')
...         y_pred = tf.matmul(w,tf.transpose(x)) +
b
...     with tf.name_scope('loss') as scope:
...         loss =
tf.reduce_mean(tf.square(y_true-y_pred))
...     with tf.name_scope('train') as scope:
...         learning_rate = 0.5
...         optimizer =
tf.train.GradientDescentOptimizer(learning_rate)
...         train = optimizer.minimize(loss)
...     # Initialisierung der Variablen vor Beginn. Wird
als Erstes ausgeführt.
...     init = tf.global_variables_initializer()
...     with tf.Session() as sess:
...         sess.run(init)
...         for step in range(NUM_STEPS):
...             sess.run(train,{x: x_data, y_true:
y_data})
...             if step % 5 == 0:
...                 print(step, sess.run([w,b]))
...                 wb_.append(sess.run([w,b]))
...         print(10, sess.run([w,b]))
...
0 [array([[0.2779293 , 0.5024658 ,
0.08615577]], dtype=float32), -0.2009458]
5 [array([[0.30149707, 0.4996172 ,
```

```
0.09656399]], dtype=float32), -0.20165265]
10 [array([[0.30149713, 0.4996172 ,
0.09656397]], dtype=float32), -0.20165262]

>>> print(x_data)
[[-0.58111234  0.83820954  1.5950554 ]
 [-0.0136423  -1.27946206  0.10544973]
 [-0.22532237  1.35862731 -0.9139968 ]
 ...
 [ 0.24275082 -0.76875081  0.10545472]
 [-0.65309978  0.38736633  0.73586307]
 [ 1.96254475  0.25914792 -0.09957582]]

>>> print(y_data)
[[ 0.16173649 -0.72014297  0.31910246 ...
-0.62691472  0.01518871
   0.46835219]]

>>> print(noise)
[[-0.04254011  0.11313578 -0.00121481 ...
-0.12591003  0.14384917
  -0.04002761]]

>>> # plot the results
... realy_data = np.matmul(w_real,x_data.T) +
b_real
>>> predictedy_data = np.matmul(w_real,x_data.T) +
b_real + noise
>>> plt.plot(realy_data, predictedy_data, 'bo',
```

```
label='Predicted y_data')
>>> plt.plot(realy_data, realy_data, 'ro', label='Real
y_data')
>>> plt.legend()
<matplotlib.legend.Legend object at 0x7fda66134cc0>
>>> plt.show()
```

Anlage:

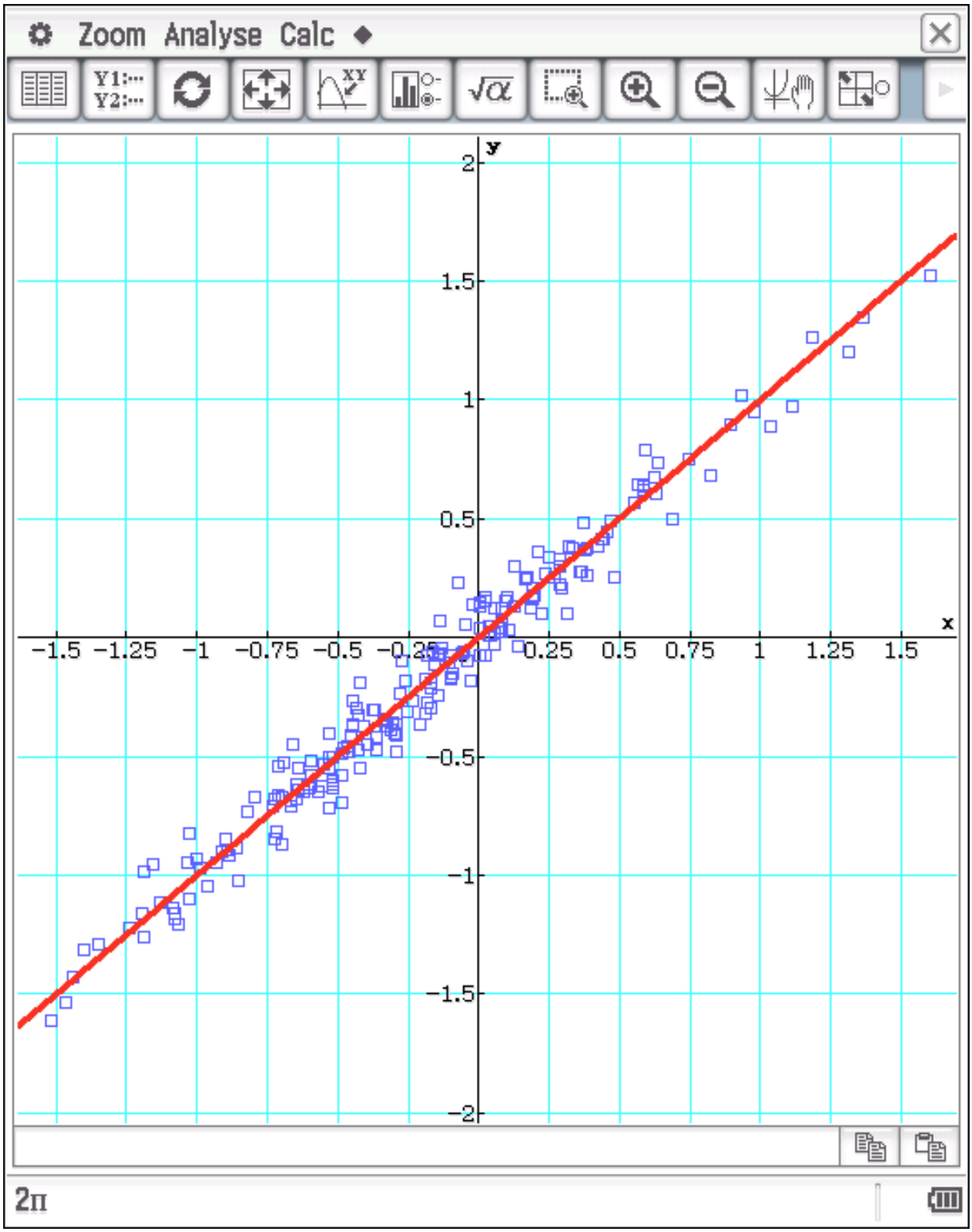
Bild mit verrauschten Daten bei linearer Regression

Download für dieses Dokument:

www.informatik.htw-dresden.de/
~paditz/Tensorflow-Ue10.pdf

Multilineare Regression mit verrauschten Daten

Datensimulation mit ClassPad: 200 Daten



Multilineare Regression mit verrauschten Daten

Datensimulation mit tensorflow: 2000 Daten