Tensor Tutorial

Lesson 4

- From Math to Programmer to Tensorflow
  - Linear machine learning example

Tensor

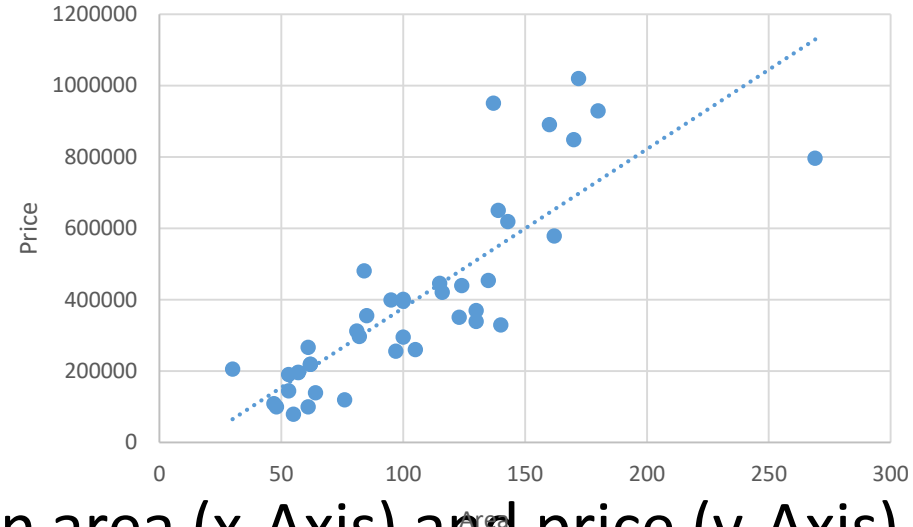
- A tensor describes a mathematical object in linear algebra
    - In programming it would be called an array
- [5,4]
    - Programer: „Array with 1 Dimension“
    - Mathematician: „Vector“
    - Tensorflow: „Tensor Rank 1“

Tensor-Overview

Example	Tensorflow	Descrption
8	Rank 0	Scalar with Shape []
[5, 8, 123]	Rank 1	Vector with Shape[3]
[[5,8,123],[100,123,1]]	Rank 2	Matrix with Shape[2,3]
[[[5,8,123]],[[100,123,1]]]	Rank 3	Tensor with Shape [2,1,3]

Estimate price of flat

- Excel table given with area and pricing



- Relation between area (x-Axis) and price (y-Axis)
  - Linear! So it can be described with a typical linear formula like:
    - F(x) = mx + b → For machine learning: y = W+x + b
    - W = „weights“
    - b = „bias“

Read out Excel files with Python

→ Use xlrd

```
import xlrd
import numpy as np
import tensorflow as tf

#Read data
def readData(filename='prices.xls'):
    b = xlrd.open_workbook(filename,encoding_override="utf-8")
    sheet = b.sheet_by_index(0)
    x = np.asarray([sheet.cell(i,1).value for i in range(1,sheet.nrows)])
    y = np.asarray([sheet.cell(i,2).value for i in range(1,sheet.nrows)])
    return x,y
```

Read out Excel files with Python

→ Use xlrd

```
import xlrd
import numpy as np
import tensorflow as tf

#Read data
def readData(filename='prices.xls'):
    b = xlrd.open_workbook(filename,encoding_override="utf-8")
    sheet = b.sheet_by_index(0)
```

An Excel file consists of more then one sheet, here you can specify the sheet you need

Read out Excel files with Python

→ Use xlrd

```
import xlrd
import numpy as np
import tensorflow as tf

#Read data
def readData(filename='prices.xls'):
    b = xlrd.open_workbook(filename,encoding_override="utf-8")
```

Specifies the column

Read out Excel files with Python

→ Use xlrd

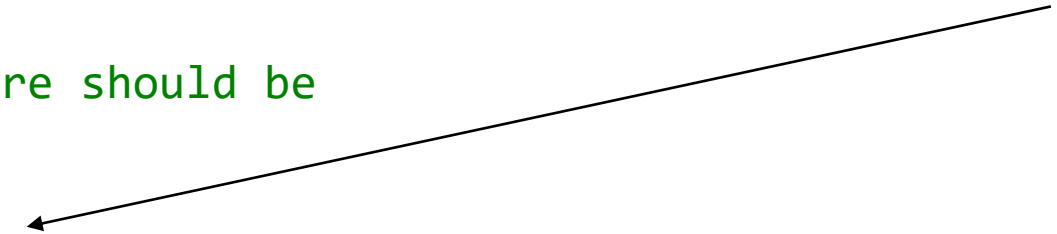
```
import xlrd
import numpy as np
import tensorflow as tf

#Read data
def readData(filename='prices.xls'):
    b = xlrd.open_workbook(filename,encoding_override="utf-8")
    sheet = b.sheet_by_index(0)
    x = np.asarray([sheet.cell(i,1).value for i in range(1,sheet.nrows)])
```

X is the input, y is the desired output

Define the model to train

```
#Read the data
x_d,y_d=readData()

#Define the model - since there should be
#a linear dependency,
#one layer should be enough
with tf.name_scope('myFirstModel'):
    #Weights, initialzed with 0
    W = tf.Variable([0.0],name='Weights')
    #bias initialzed with 0
    b = tf.Variable([0.0],name='bias')
    y = W*x_d + b
```

Sum up everything of the computation into one scope
→ TensorBuild now shows this as only one node



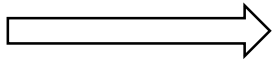
Define the training

- Training means finding the minimum distance of the estimated value to the ground truth and change the weights accordingly → Loss

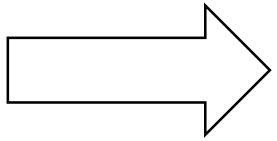
```
#Define the graph for training
with tf.name_scope('myFirstTraining'):
    #Calculate loss
    l = tf.reduce_mean(tf.square(y - y_d),name='loss')

    tf.summary.scalar('loss',l)


    #Optimizing Algorithm
    opt = tf.train.GradientDescentOptimizer(0.00001)
    t = opt.minimize(l)
```

Training

```
#Define the graph for training
with tf.name_scope('myFirstTraining'):
    #Calculate loss
    l = tf.reduce_mean(tf.square(y - y_d),name='loss')

    tf.summary.scalar('loss',l)


    #Optimizing Algorithm
    opt = tf.train.GradientDescentOptimizer(0.00001)
    t = opt.minimize(l)
```



Calculation of the average error over the whole vector („loss“)
→ The smaller the better!



Calculate the derivation to estimate how to update the weights with the parameter learning rate

Run the session

```
#learn!
summary_of_ops = tf.summary.merge_all()
print(summary_of_ops)
```

Summary of Operations
for every training iteration

```
with tf.Session() as session:
    wr = tf.summary.FileWriter('./housing',session.graph)
    init = tf.global_variables_initializer()
    session.run(init)

    #Train
    for i in range(100000):
        print(i)
        summary,_ = session.run([summary_of_ops,t])
        wr.add_summary(summary,i)

    #Test
    cW,cB,cl = session.run([W,b,l])
    print("Weights: %s bias: %s loss: %s" % (cW,cB,cl))

wr.close()
```