TensorFlow intro

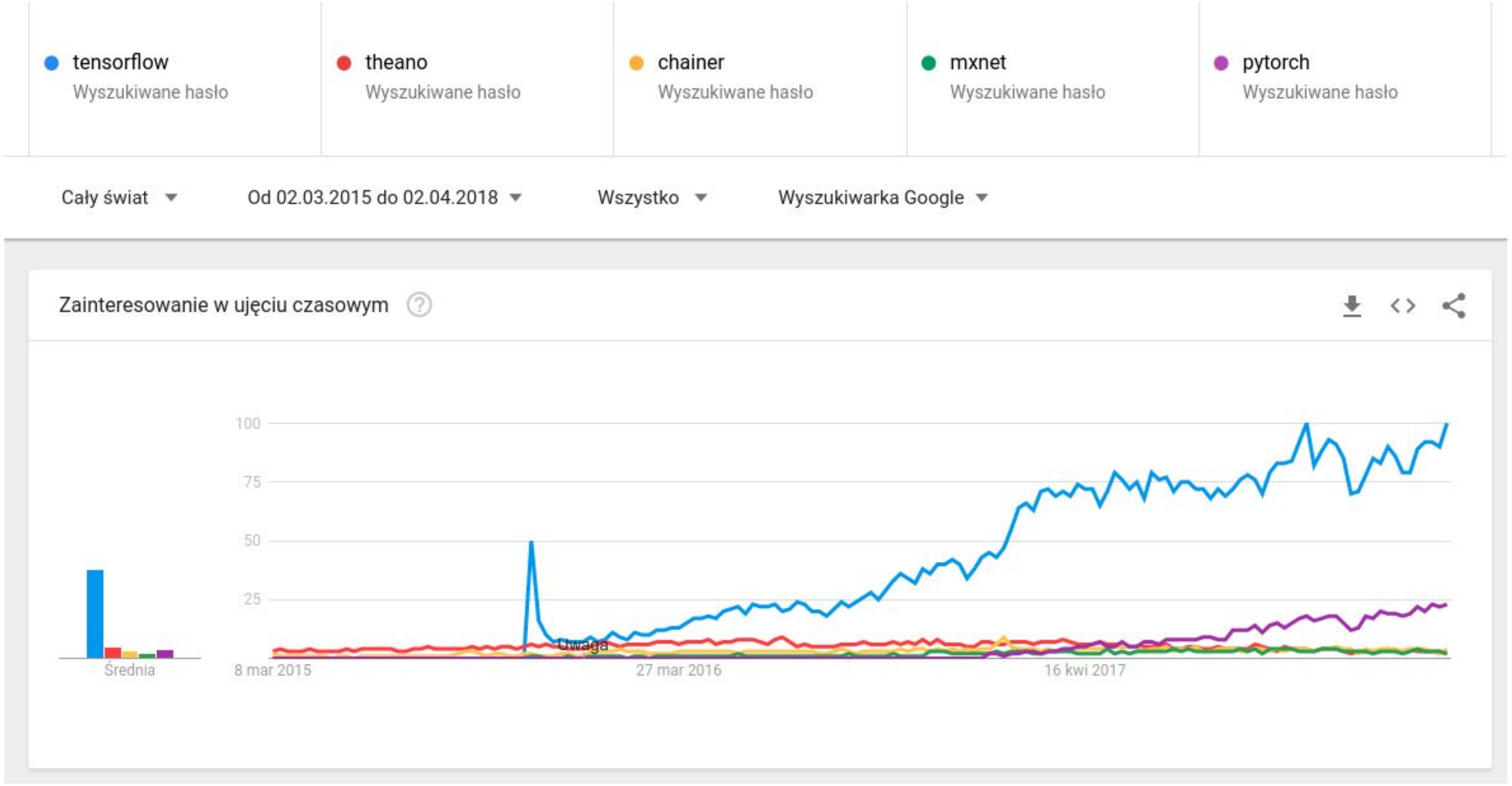
TensorFlow

Maciej Klimek

What is Tensorflow

- TensorFlow™ is an open source software library for high performance numerical computation (particularly GPUs)
- Strong support for machine learning and deep learning(like gradient computations)

The most popular ML framework today



● tensorflow Wyszukiwane hasło
● theano Wyszukiwane hasło
● chainer Wyszukiwane hasło
● mxnet Wyszukiwane hasło
● pytorch Wyszukiwane hasło
Cały świat
Od 02.03.2015 do 02.04.2018
Wszystko
Wyszukiwarka Google
Zainteresowanie w ujęciu czasowym
100
75
50
25
Średnia
8 mar 2015
27 mar 2016
16 kwi 2017
Uwaga

The most popular ML framework today

airbnb AMD NVIDIA. UBER

SAP kakao DeepMind Dropbox

ebay Google intel

Coca-Cola mi ZTE QUALCOMM

Hello World

```
import tensorflow as tf

sess = tf.InteractiveSession()

a = tf.constant(5.0)

b = tf.constant(6.0)

c = a * b

print(c.eval())
```

```
30.0
```

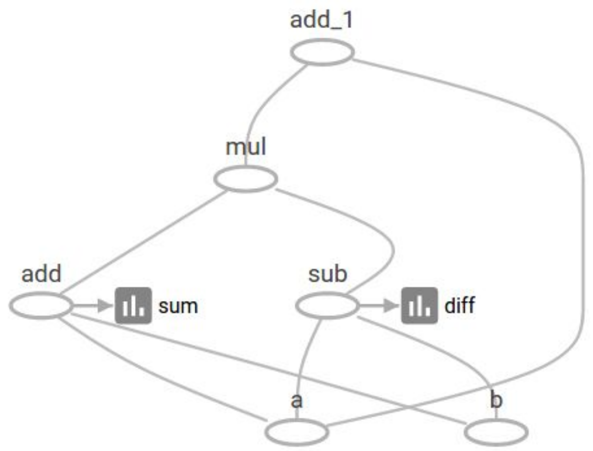
Hello World

```
import tensorflow as tf
sess = tf.InteractiveSession() # What is this?
a = tf.constant(5.0)
b = tf.constant(6.0)
c = a * b
print(c.eval()) # Why do we have to do this, can’t we just print(c),
to get result?

30.0
```

Computation graphs

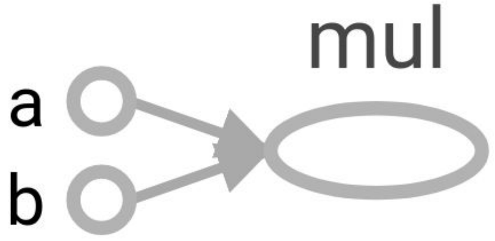
- “TensorFlow programs are usually structured into a construction phase, that assembles a graph, and an execution phase that uses a session to execute ops in the graph.” - TensorFlow docs
- A computation graph is a DAG. It has “mathematical” operations as nodes(also called ops)
  and tensor(multi-dimensional array as edges)
- Tensors are flowing through the computation graph, hence the name TensorFlow

-
  ```
  pip search tensorflow
  tensorflow (1.7.0) - TensorFlow helps the tensors flow
  ```



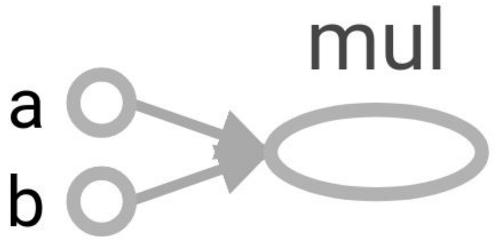
Hello World revisited

```
 import tensorflow as tf

 # Creating the computation graph phase
a = tf.constant(5.0, name=’a’)
b = tf.constant(6.0, name=’b’)
 # type(a) == <class 'tensorflow.python.framework.ops.Tensor'>

# We are just creating a node in the computation graph here, no 5.0 * 6.0,
m = a * b # same as m = tf.multiply(a, b)

# type(m) == <class 'tensorflow.python.framework.ops.Tensor'>
```



Hello World revisited

```
import tensorflow as tf
# Creating the computation graph phase
a = tf.constant(5.0, name='a')
b = tf.constant(6.0, name='b')

# We are just creating a node in the computation graph here, no 5.0 * 6.0
m = a * b

# Running the computation graph phase
with tf.Session() as sess:
    print(m.eval()) # result = 30.0
    print(a.eval()) # result = 5.0
    print(sess.run([m, a])) # result = [30.0, 5.0]
```



Hello World 2.0

```
import tensorflow as tf
sess = tf.InteractiveSession()
a = tf.placeholder(dtype=tf.float32, name=’a’)
b = tf.placeholder(dtype=tf.float32, name=’b’)
c = a * b
feed_dict = {a: 5.0, b: 6.0}
print(sess.run(c, feed_dict))
```

- Use tf.placeholder variables (dummy nodes that provide entry points for data to computational graph).
- A feed_dict is a python dictionary mapping from tf. placeholder vars (or their names) to data (numpy arrays, lists, etc.).

What is wrong?

```
import tensorflow as tf
sess = tf.InteractiveSession()
a = tf.placeholder(dtype=tf.float32, name='a')
b = tf.placeholder(dtype=tf.float32, name='b')
X = [0.0, 1.0, 5.0, 5.0]
Y = [2.0, 3.0, 1.0, 8.0]
for x, y in zip(X, Y):
    feed_dict = {a: x, b: y}
    print(sess.run(a + b, feed_dict))
```

tf.Variable

```
import tensorflow as tf
sess = tf.InteractiveSession()
a = tf.Variable(0.0, dtype=tf.float32, name='a')

value_to_assign = tf.placeholder(dtype=tf.float32)
assign_op = a.assign(value_to_assign)

# print a.eval() # this would cause error because variable a is not initialized yet.
init_op = tf.global_variables_initializer()
sess.run(init_op)

print(a.eval()) # result = 0.0
sess.run(assign_op, feed_dict={value_to_assign: 3.0})
print(a.eval()) # result = 3.0
```

tf.Variable

```
import tensorflow as tf
sess = tf.InteractiveSession()
b = tf.Variable(tf.random_uniform([2, 2], -0.1, 0.1), dtype=tf.float32, name='b')
c = tf.Variable(tf.random_normal([2, 2], stddev=0.1), dtype=tf.float32, name='c')

init_op = tf.global_variables_initializer()
sess.run(init_op)
print(b.eval())
print(c.eval())
```

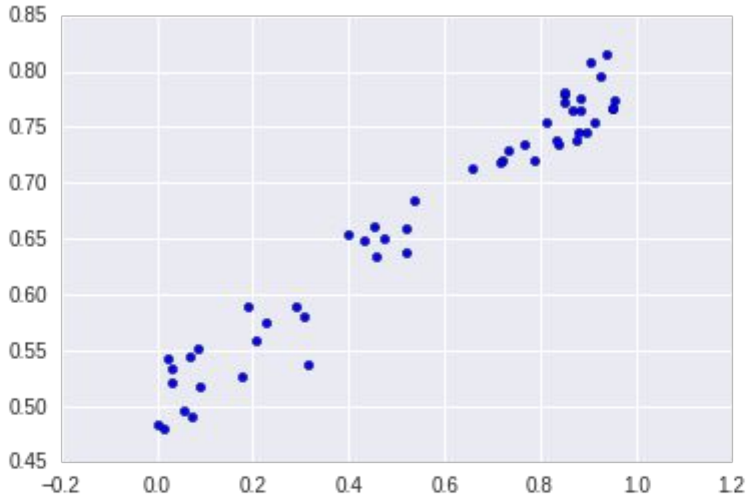
```
[[-0.01472919  0.08135603]
 [ 0.07122798  0.03257368]]
[[0.03300044 0.1379751 ]
 [0.08122743 0.06315189]]
```

tf.gradients

```
import tensorflow as tf
x = tf.placeholder(dtype=tf.float32)
f_x = x * x
df_dx = tf.gradients(xs=x, ys=f_x)

with tf.Session() as sess:
    feed_dict = {x: 3.0}
    print(sess.run([x, f_x, df_dx], feed_dict))

# Result: [array(3., dtype=float32), 9.0, [6.0]]
```

Linear Regression

```
import numpy as np
import seaborn
import matplotlib.pyplot as plt
import random



f = lambda x: 0.3 * x + 0.5 # ground truth
g = lambda x: f(x) + random.gauss(0, 0.02) # a noisy version of f

X_data = [random.random() for _ in xrange(50)]
y_data = map(g, X_data)

plt.scatter(X_data, y_data)

X_data = np.reshape(X_data, (50, 1))
y_data = np.reshape(y_data, (50, 1))
```



Linear Regression

```
# Creating placeholders for input
X = tf.placeholder(shape=(None, 1), dtype=tf.float32)
y_gt = tf.placeholder(shape=(None, 1), dtype=tf.float32)

# Creating model
W = tf.Variable(np.zeros((1, 1), dtype='float32'), name='W')
b = tf.Variable(0.0, name='b')
y_pred = tf.matmul(X, W) + b
loss = tf.reduce_mean((y_gt - y_pred) ** 2)
```

Linear Regression

```
vars_to_train = [W, b]
gradients = tf.gradients(xs=vars_to_train, ys=loss)
assign_ops = []
for var, grad in zip(vars_to_train, gradients):
    assign_ops.append(var.assign(var - grad * 0.1))
gradient_step = assign_ops
```

Linear Regression

```
vars_to_train = [W, b]
gradients = tf.gradients(xs=vars_to_train, ys=loss)
assign_ops = []
for var, grad in zip(vars_to_train, gradients):
    assign_ops.append(var.assign(var - grad * 0.1))
gradient_step = assign_ops


# OR using built-in GradientDescentOptimizer

optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1)
gradient_step = optimizer.minimize(loss)
```

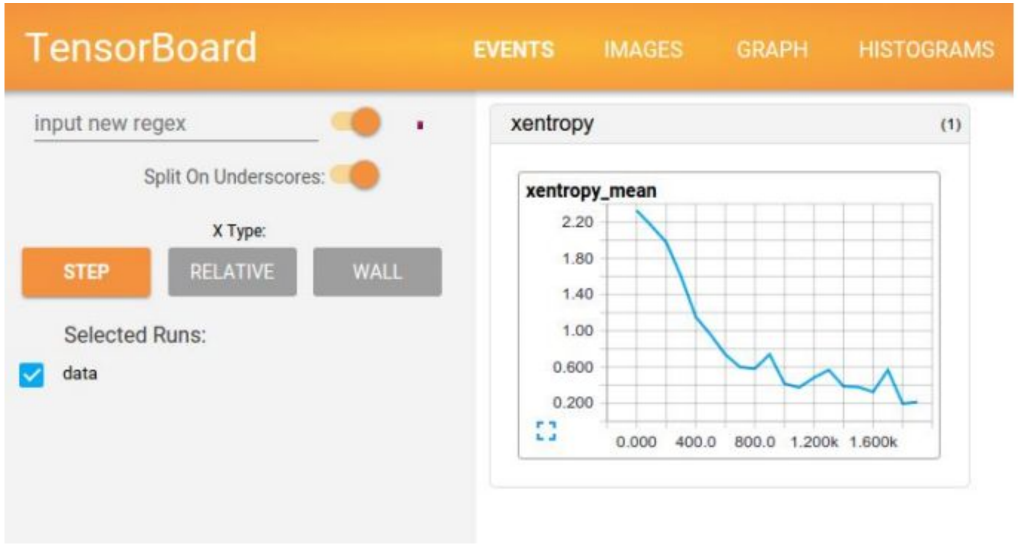
Linear Regression

```
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for epoch_idx in range(1000):
        feed_dict = {X: X_data, y_gt: y_data}
        _, loss_value = sess.run([gradient_step, loss], feed_dict)
        #print epoch_idx, loss_value

    print(sess.run(W), sess.run(b))
```

```
[[ 0.28261772]] 0.509359
```

Tensorboard

- Track progress of learning
- Compare experiments
- Save images/audio clips



Questions?

TensorFlow Lab

Labs

Work through the tutorials at your own pace:

https://www.tensorflow.org/versions/r1.4/get_started/get_started (don’t read part about `tf.estimator`) We are using tf version 1.4 because latter versions expose more high-level API, which hide how things actually work.
https://www.tensorflow.org/versions/r1.4/get_started/mnist/beginners

Don’t just read the code, try to run it simultaneously in the notebook/editor as you go through the text.

There are two task you can try solving:

- exc1.ipynb - some introductory exercises
- mnist_tf.py - train your model on MNIST dataset

It is best to work in virtualenv.
To use jupyter with a virtualenv just install it in this virtualenv:‘pip install jupyter’, and run ‘jupyter notebook’ having the virtualenv already activated.

virtualenvwrapper (https://virtualenvwrapper.readthedocs.io/en/latest) is very convenient for managing virtualenvs.
Exercises work in python3.