TensorFlow konvolúciós hálózatokhoz

Aradi Bernadett

2017/18 ősz

Tensorflow

import tensorflow as tf

- szoftverkönyvtár neurális hálózatokhoz
- a Google Brain csapata hozta létre, open-source (2015 nov. óta)
- hasonló szoftverek: Caffe (UC Berkeley), Theano (Université de Montréal)
- segítségével a számítások eloszthatóak CPU-kon, GPU-kon, különböző eszközökön
- számítások vizualizációja: TensorBoard használatával
  Data flow graphs:
  1. gráf csomópontja: matematikai műveletek
  2. gráf élei: adattömbök (tenzorok)
- C++ backend, ezzel való kapcsolat: „Session”-ön keresztül
  A Tensorflow különválasztja a számítások definiálását a végrehajtásuktól!
  1. „graph” megkonstruálása (TF látja, mit kell Pythonon kívül csinálni)
  2. „launch the graph in a session” (lefuttatjuk a graph-ot)

Tenzorok

„tensor” = multidimenziós adattömb

- 0D: szám, 1D: vektor, 2D: mátrix, ...
- tömbök definiálása: `tf.constant` paranccsal

Építsünk egy graph-ot!

- `matrix1 = tf.constant([[2.],[2.]])` # 2x1-es mátrix
- `matrix2 = tf.constant([[3.,3,3]])` # 1x3-as mátrix
- `szorzat = tf.matmul(matrix1, matrix2)` # mátrixszorzás
- `print(szorzat)` # mit kapunk?

Futtassuk le a graph-ot egy session-ben!

- `sess = tf.Session()` # Session: környezet, ahol
  #az operátorokat végrehajtjuk, tenzorokat kiértékeljük
- `eredmeny = sess.run(szorzat)`
  `print(eredmeny)` # eredmény: numpy ndarray formátumú
  `sess.close()` # lezárjuk a munkamenetet

Vizualizáció TensorBoardban

```
tensorboard --logdir='/your/path/here'
tensorboard --logdir=training:C:/Users/.../graphs
```

```
import tensorflow as tf
a = tf.constant(2)
b = tf.constant(3)
x = tf.add(a,b)

with tf.Session() as sess:
# graph definiálása: session futtatása előtt
    writer = tf.summary.FileWriter('./graphs', sess.graph)
    print(sess.run(x))
writer.close()
```

Megkönnyítjük a helyzetünket, ha mi nevezzük el a változókat:

```
a = tf.constant(2,name='a')
b = tf.constant(3,name='b')
x = tf.add(a,b,name='osszeg')
```

tf.constant máshogy

```
a = tf.zeros([2, 3], tf.int32)       # nullákkal kitöltés

input = tf.constant([[0, 1], [2, 3]])
b = tf.zeros_like(input)

c = tf.fill([2, 3], 8)       # számmal kitöltés

d = tf.linspace(10.0, 13.0, 4)       # számok egyenlő távolságra
e = tf.range(5)

f = tf.random_normal([2, 3])       # véletlen számok normális eloszlásból
with tf.Session() as sess:
    print(sess.run(a))
    print(sess.run(b))
    print(sess.run(c))
    print(sess.run(d))
    print(sess.run(e))
    print(sess.run(f))
```

tf.Variable – a tanítandó változók

```
a = tf.Variable(2, name="scalar")        # szám
b = tf.Variable([2, 3], name="vector")       # vektor
c = tf.Variable([[0, 1], [2, 3]], name="matrix")       # mátrix
d = tf.Variable(tf.ones([2, 3]))       # mátrix egyesekkel kitöltve

# a változókat inicializálni kell használat előtt!

init = tf.global_variables_initializer()   # az összes változó inicializálása
with tf.Session() as sess:
    sess.run(init)

init_ab = tf.variables_initializer([a, b])      # néhány változó inicializálása
with tf.Session() as sess:
    sess.run(init_ab)

with tf.Session() as sess:        # egy változó inicializálása
    sess.run(d.initializer)
    print(d)       # ez még nem az érték
    print(d.eval())       # ez igen, ugyanaz mint: print(sess.run(d))
```

Session jellegzetességei

```
a = tf.Variable(10)
a.assign(100)       # assign: hozzárendelés
with tf.Session() as sess:
    sess.run(a.initializer)
    print(a.eval())
```

10 maradt az értéke, mert az assign-t is le kell futtatni a sessionben!

```
a = tf.Variable(10)
a2 = a.assign(100)
with tf.Session() as sess:
    # sess.run(a.initializer)
    # ez a sor feleslegessé válik, mert az assign inicializálja a változót
    sess.run(a2)
    print(a.eval())
```

Session jellegzetességei 2.

Minden session fenntartja a saját változatát a változóból!

```
a = tf.Variable(10)

sess1 = tf.Session()
sess2 = tf.Session()

sess1.run(a.initializer)
sess2.run(a.initializer)

print(sess1.run(a.assign_add(10)))
print(sess2.run(a.assign_sub(2)))

print(sess1.run(a.assign_add(100)))
print(sess2.run(a.assign_sub(50)))

sess1.close()
sess2.close()
```

tf.placeholder – a még ismeretlen adat

1. létrehozzuk a graphot
2. egy sessionben lefuttatjuk a graph operációit

Összerakhatjuk a graphot úgy, hogy még nem tudjuk az értékeket!

```
a = tf.placeholder(tf.float32, shape=[3])    # placeholder, 3 elemű vektor
b = tf.constant([5, 5, 5], tf.float32)       # ugyanolyan alakú konstans
c = a + b

with tf.Session() as sess:
    # print(sess.run(c))       # ez így hibát dob, hiányzik az érték
    print(sess.run(c, feed_dict={a: [1, 2, 3]}))  # „meg kell etetni" az a-t
```

```
a = tf.placeholder(tf.float32)
b = tf.constant(10.0)
c = a + b
list = [1, 2, 3]       # több értékkel is megetethetjük a placeholdert

with tf.Session() as sess:
    for values in list:
        print(sess.run(c, feed_dict={a: values}))
```

Általános struktúra modellépítéshez

Graph összerakása

1. Adatok beolvasása (placeholderek az inputok és outputok számára)
2. Súlyok definiálása (variables)
3. Modell definiálása
4. Veszteségfüggvény definiálása
5. Optimalizáló függvény definiálása

Számítás

1. Session létrehozása
2. Változók inicializálása
3. Tanítás (ez lefuttatja a becslést, kiszámítja a gradienseket és megteszi a szükséges változtatásokat a hiba minimalizálásához)
4. Szükséges értékek lekérdezése

Optimalizáló eljárások:

`https://www.tensorflow.org/api_guides/python/train#optimizers`

Rétegek:

`https://www.tensorflow.org/api_docs/python/tf/layers`