Keras

kertesz.gabor@nik.uni-obuda.hu

Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation.
Being able to go from idea to result with the least possible delay is key to doing good research.

https://keras.io/

Keras

Francois Chollet, a Google mérnöke fejleszti. 2017 óta a TensorFlow csomag részeként is elérhető.

Fő irányadó elvek:

- Felhasználóbarátság
- Modularitás
- Egyszerű bővíthetőség
- Python

30 seconds to… Keras

A Keras alapeleme a Model, amelyben a bemenettől a kimenetig rétegeket definiálunk.

Az alapvető Model típus a Kerasban a Sequential, de lehetőség van bővítésre is.

A Modelhez rétegek hozzáadása az .add() metódussal egyszerűen megoldható:

```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(units=64, activation='relu', input_dim=100))
model.add(Dense(units=10, activation='softmax'))
```

30 seconds to… Keras

A loss és optimizer egyszerűen konfigurálható – ezt követően indulhat a tanítás:

```
model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5, batch_size=32)
```

Emlékeztető: TensorFlow binary classification

```
X = tf.placeholder(tf.float32, [None, input_params])
Y = tf.placeholder(tf.float32, [None, 1])

W1 = tf.Variable(tf.random_normal([input_params, 20]), name='W1')
b1 = tf.Variable(tf.random_normal([20]), name='b1')
W2 = tf.Variable(tf.random_normal([20, 8]), name='W2')
b2 = tf.Variable(tf.random_normal([8]), name='b2')
W3 = tf.Variable(tf.random_normal([8, 1]), name='W3')
b3 = tf.Variable(tf.random_normal([1]), name='b3')

Z1 = tf.add(tf.matmul(X, W1), b1)
A1 = tf.nn.relu(Z1)
#A1 = tf.nn.sigmoid(Z1)
Z2 = tf.add(tf.matmul(A1, W2), b2)
A2 = tf.nn.relu(Z2)
#A2 = tf.nn.sigmoid(Z2)
Z3 = tf.add(tf.matmul(A2, W3), b3)

sigm = tf.nn.sigmoid(Z3, name='sigm')
loss = tf.reduce_mean(tf.nn.sigmoid_cross_entropy_with_logits(logits=Z3,
labels=Y))
pred = tf.cast(tf.greater_equal(sigm, 0.5), tf.float32, name='pred')
acc = tf.reduce_mean(tf.cast(tf.equal(pred, Y), tf.float32), name='acc')

optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(loss)

init = tf.global_variables_initializer()
```

```
with tf.Session() as sess:
    sess.run(init)

    for epoch in range(training_epochs):
        sess.run(optimizer, feed_dict={X: train_X, Y: train_Y})

        if (epoch + 1) % display_step == 0:
            c = sess.run(loss, feed_dict={X: train_X, Y: train_Y})
            train_acc = sess.run(acc, feed_dict={X: train_X, Y: train_Y})
            test_acc = sess.run(acc, feed_dict={X: test_X, Y: test_Y})
            print("Epoch:", '%04d' % (epoch + 1), "cost=",
"{:.9f}".format(c),
                  "train_acc {:.2f}%".format(100 * train_acc), "test_acc
{:.2f}%".format(100 * test_acc))

    print("Tanitas vege!")
    training_cost = sess.run(loss, feed_dict={X: train_X, Y: train_Y})
    test_preds = sess.run(pred, feed_dict={X: test_X, Y: test_Y})
```

Szemtől szemben: TF vs Keras

```
X = tf.placeholder(tf.float32, [None, input_params])
Y = tf.placeholder(tf.float32, [None, 1])

W1 = tf.Variable(tf.random_normal([input_params, 20]),
name='W1')
b1 = tf.Variable(tf.random_normal([20]), name='b1')
W2 = tf.Variable(tf.random_normal([20, 8]), name='W2')
b2 = tf.Variable(tf.random_normal([8]), name='b2')
W3 = tf.Variable(tf.random_normal([8, 1]), name='W3')
b3 = tf.Variable(tf.random_normal([1]), name='b3')

Z1 = tf.add(tf.matmul(X, W1), b1)
A1 = tf.nn.relu(Z1)
Z2 = tf.add(tf.matmul(A1, W2), b2)
A2 = tf.nn.relu(Z2)
Z3 = tf.add(tf.matmul(A2, W3), b3)
```

```
model = Sequential()
model.add(Dense(20, activation='relu',
input_shape=(input_params, )))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

Szemtől szemben: TF vs Keras

```
sigm = tf.nn.sigmoid(Z3, name='sigm')
loss = tf.reduce_mean(tf.nn.sigmoid_cross_entropy_with_logits(logits=Z3, labels=Y))
pred = tf.cast(tf.greater_equal(sigm, 0.5), tf.float32, name='pred')
acc = tf.reduce_mean(tf.cast(tf.equal(pred, Y), tf.float32), name='acc')

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

```
model.compile(
    optimizer=tf.train.GradientDescentOptimizer(learning_rate=learning_rate),
    loss='binary_crossentropy', metrics=['accuracy'])
```

Szemtől szemben: TF vs Keras

```
init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)

    for epoch in range(training_epochs):
        sess.run(optimizer, feed_dict={X: train_X, Y: train_Y})
```

```
model.fit(train_X, train_Y, epochs=training_epochs, verbose=2,
validation_data=(test_X, test_Y))
```

Jupyter notebook

- Parancssorban: `jupyter notebook`
- Az elindult webszerverre böngészővel navigálhatunk, ahol az úgynevezett Jupyter dokumentumok megtekinthetőek, illetve a beágyazott Python kód interaktívan futtatható

https://jupyter.readthedocs.io/en/latest/running.html