

פעילות 14 - מימוש שער XOR על ידי מכונה לומדת תחת Keras

מקורות:

<https://github.com/penseeartificielle/perceptron-xor-examples/blob/master/keras-xor.py>

<https://gist.github.com/stewartpark/187895beb89f0a1b3a54>

<https://victorzhou.com/blog/keras-neural-network-tutorial/>

Keras היא ספריית נוספת לפיתוח רשתות נוירונים מלאכותיות. הספרייה נכתבה בשפת Python כחלק מקוד פתוח. בשנת 2017 החליטו צוות מפתחי TensorFlow של גוגל לתמוך בקוד הפתוח של Keras. משמעות הדבר היא שהספרייה Keras יושבת מעל הספרייה של TensorFlow ומספקת ממשק גישה נוח לספרייה TensorFlow. לפי הערכתי TensorFlow מהווה את חוד החנית של הטכנולוגיה לפיתוח רשתות נוירונים. בפעילות זו נתרגל כתיבת קוד תוך שימוש בכלי Keras לפיתוח ANN – Artificial Neural Network.

התקנות

התקנת הספרייה תבצע על ידי ההוראה הבאה:

```
> python --version
> python -m pip install -U pip
> python -m pip install -U scipy
> python -m pip install -U mnist
> python -m pip install -U numpy
> python -m pip install -U tensorflow
> python -m pip install -U keras
> python -m pip install -U nltk
```

כדי לבדוק את גרסה tensorflow שהתקנתם ניתן לכתוב את ההוראות הבאות:

```
> python -c 'import tensorflow as tf; print(tf.__version__)'
> python -c 'import keras; print(keras.__version__)'
```

נקבל את הפלט הבא:

```
PS C:\Users\gadi> python -c 'import tensorflow as tf; print(tf.__version__)'
2020-02-29 16:39:10.718906: W tensorflow/stream_executor/platform/default/dso
ry 'cudart64_101.dll'; dlerror: cudart64_101.dll not found
2020-02-29 16:39:10.742956: I tensorflow/stream_executor/cuda/cudart_stub.cc:
ot have a GPU set up on your machine.
2.1.0
PS C:\Users\gadi> python -c 'import keras; print(keras.__version__)'
Using TensorFlow backend.
2020-02-29 16:42:05.661348: W tensorflow/stream_executor/platform/default/dso
ry 'cudart64_101.dll'; dlerror: cudart64_101.dll not found
2020-02-29 16:42:05.666217: I tensorflow/stream_executor/cuda/cudart_stub.cc:
ot have a GPU set up on your machine.
2.3.1
```

תוכנית ראשונה ב- Keras למימוש שער XOR

נדגים את העקרונות בתכנות רשתות נוירונים תוך שימוש בדוגמא שכבר הכרנו לפני ולפנים.

המרכיב הבסיסי של רשת נוירונים מבוססת Keras היא model

```
from keras.models import Model
```

קיימים שני סוגים של מודלים:

- Sequential model - מודל סדרתי שבו קיים רצף לינארי של שכבות (כלומר כל מוצא של שכבה מחוברת למבוא של השכבה הבאה).
- Functional Model - מודל המבוסס על יכולת לבנות רשתות נוירונים שאין להם רצף לינארי, לדוגמה רשת הכוללת מספר מקורות קלט שונים, מספר מקורות פלט שונים במאפיינים שלהם או רשתות הכוללת קטעים חבויים החוזרים על עצמם מספר פעמים.

לצורך מימוש רשת נוירונים לסיווג XOR הגדרנו רשת סדרתית לינארית הכוללת 2 נוירונים בשכבת המבוא 8 נוירונים בשכבה הבאה שבמוצא של כל אחד מהם פונקציית המעבר היא מסוג tanh כמו כן הגדרנו שכבת מוצא אחת הכוללת פונקציית מעבר מסוג sigmoid.

```
model = Sequential()
model.add(Dense(8, input_dim=2))
model.add(Activation('tanh'))
model.add(Dense(1))
model.add(Activation('sigmoid'))
```

כתבו את הקוד הבא והריצו אותו:

```
from keras.models import Sequential
from keras.layers.core import Dense, Activation

model = Sequential()
model.add(Dense(8, input_dim=2))
model.add(Activation("tanh"))
model.add(Dense(1))
model.add(Activation("sigmoid"))

print(model.summary())
```

נקבל את הפלט הבא:

```
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 8)	24
activation_1 (Activation)	(None, 8)	0
dense_2 (Dense)	(None, 1)	9
activation_2 (Activation)	(None, 1)	0

```

Total params: 33
Trainable params: 33
Non-trainable params: 0

```

מפלט התוכנית אנו למדים על מבנה הרשת הכוללת שה"כ 33 קשרים בין הפרספטרונים. אופן חישוב הקשרים מתבצע כך:

$$\text{מספר המוצאים} * (\text{מספר המבואות} + 1)$$

$$8 * (2 + 1) = 24$$

$$1 * (8 + 1) = 9$$

סה"כ 33

השלב הבא יהיה לזמן את הפעולה compile השייכת למחלקה model. פעולה זו מגדירה את מודל האימון של המכונה. נדגים זאת:

```
sgd = SGD(lr=0.1)
model.compile(loss='binary_crossentropy', optimizer=sgd)
```

השלב האחרון יהיה הפעלת הפעולה שמבצעת את האימון בפועל.

```
model.fit(X, y, batch_size=1, nb_epoch=500)
```

להלן קוד התוכנית המממש שער XOR על ידי מכונה לומדת תוך שימוש בספרייה של keras לבניית רשת הניורונים

```

from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation
from keras.optimizers import SGD
import numpy as np

X = np.array([[0,0],[0,1],[1,0],[1,1]])
y = np.array([[0],[1],[1],[0]])

```

```
model = Sequential()
model.add(Dense(8, input_dim=2))
model.add(Activation('tanh'))
model.add(Dense(1))
model.add(Activation('sigmoid'))

sgd = SGD(lr=0.1)
model.compile(loss='binary_crossentropy', optimizer=sgd)

model.fit(X, y, batch_size=1, nb_epoch=500)
print(model.predict_proba(X))
```

פלט התוכנית יהיה:

```
Epoch 499/500
4/4 [=====] - 0s 500us/step - loss: 0.0137
Epoch 500/500
4/4 [=====] - 0s 500us/step - loss: 0.0136
[[0.00549314]
 [0.9850749 ]
 [0.9860681 ]
 [0.01927511]]
```

תנאי השימוש

תנאי השימוש במסמך זה הם לפי הסטנדרט הבא:

You are free:

to Share – to copy, distribute and transmit the material
to Remix – to adapt the material

Under the following conditions:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

NonCommercial — You may not use the material for commercial purposes.

ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.