Logic Gates with Keras

August 20, 2019

1 Keras ile lojik kaplarn implementasyonu - (AND,OR,XOR,XNOR,NAND)

[7]:
```
import tensorflow as tf
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from keras.utils.vis_utils import plot_model
import os
```

2 AND Gate

[8]:
```
X = np.array([
    [0,0],
    [0,1],
    [1,0],
    [1,1]
])
and_Y = np.array([[0],[0],[0],[1]])
```

[9]:
```
AND=Sequential()
AND.add(Dense(1, input_dim=2, activation='sigmoid'))
AND.compile(loss='binary_crossentropy', optimizer='adam')
AND.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_2 (Dense)              (None, 1)                 3
=================================================================
Total params: 3
Trainable params: 3
Non-trainable params: 0
_________________________________________________________________
```

[ ]:
```
history = AND.fit(X, and_Y, epochs=1500, verbose=1)
```

[12]:
```
predictions=AND.predict(X)

predictions
```

[12]:
```
array([[0.22192268],
       [0.3714459 ],
       [0.40155524],
       [0.58163506]], dtype=float32)
```

[13]:
```
k=0
for i in predictions:

    if( i>= 0.5):
        print("input: ", X[k],"=" ,1)
    else:

        print("input: ", X[k],"=" ,0)
    k+=1
```

```
input:  [0 0] = 0
input:  [0 1] = 0
input:  [1 0] = 0
input:  [1 1] = 1
```

3 OR

[ ]:
```
or_Y = np.array([[0],[1],[1],[1]])

OR=Sequential()
OR.add(Dense(2, input_dim=2, activation='tanh'))
OR.add(Dense(1, activation='sigmoid'))

OR.compile(loss='binary_crossentropy', optimizer='adam')
print(OR.summary())
history = OR.fit(X, or_Y, epochs=1350)
```

[101]:
```
#predict
predictions=OR.predict(X)
k=0
for i in predictions:

    if( i> 0.5):
        print("input: ", X[k],"=" ,1)
    else:

        print("input: ", X[k],"=" ,0)
```

```
    k+=1
```

```
input:  [0 0] = 0
input:  [0 1] = 1
input:  [1 0] = 1
input:  [1 1] = 1
```

4 XOR

[102]:
```
xor_Y = np.array([[0],[1],[1],[0]])

XOR=Sequential()
XOR.add(Dense(4, input_dim=2, activation='relu'))
XOR.add(Dense(1, activation='sigmoid'))
XOR.compile(loss='binary_crossentropy', optimizer='adam')#min kac tane node␣
 ↪toplam olarak
print(XOR.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_62 (Dense)             (None, 4)                 12
_________________________________________________________________
dense_63 (Dense)             (None, 1)                 5
=================================================================
Total params: 17
Trainable params: 17
Non-trainable params: 0
_________________________________________________________________
None
```

[ ]:
```
history = XOR.fit(X, xor_Y, epochs=700)
```

[64]:
```
predictions=XOR.predict(X)
predictions
```

[64]:
```
array([[0.3496899 ],
       [0.6884465 ],
       [0.64192325],
       [0.34956402]], dtype=float32)
```

[65]:
```
#predict

k=0
for i in predictions:

    if( i> 0.5):
        print("input: ", X[k],"=" ,1)
```

```
    else:

        print("input: ", X[k],"=" ,0)
    k+=1
```

```
input:  [0 0] = 0
input:  [0 1] = 1
input:  [1 0] = 1
input:  [1 1] = 0
```

5 XNOR

[103]:
```
xNor_Y = np.array([[1],[0],[0],[1]])

XNOR=Sequential()
XNOR.add(Dense(4, input_dim=2, activation='relu'))
XNOR.add(Dense(1, activation='sigmoid'))
XNOR.compile(loss='binary_crossentropy', optimizer='adam',  metrics=['acc'])
print(XNOR.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_64 (Dense)             (None, 4)                 12
_________________________________________________________________
dense_65 (Dense)             (None, 1)                 5
=================================================================
Total params: 17
Trainable params: 17
Non-trainable params: 0
_________________________________________________________________
None
```

[ ]:
```
history = XNOR.fit(X, xNor_Y, epochs=1500)
```

[67]:
```
#predict
predictions=XNOR.predict(X)
k=0
for i in predictions:

    if( i> 0.5):
        print("input: ", X[k],"=" ,1)
    else:

        print("input: ", X[k],"=" ,0)
    k+=1
```

```
input:  [0 0] = 1
input:  [0 1] = 0
input:  [1 0] = 0
input:  [1 1] = 1
```

6 NAND

[104]:
```
nand_Y = np.array([[1],[1],[1],[0]])

NAND=Sequential()
NAND.add(Dense(1, input_dim=2, activation='sigmoid'))
NAND.compile(loss='binary_crossentropy', optimizer='adam')
print(NAND.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_66 (Dense)             (None, 1)                 3
=================================================================
Total params: 3
Trainable params: 3
Non-trainable params: 0
_________________________________________________________________
None
```

[ ]:
```
history = NAND.fit(X, nand_Y, epochs=1500)
```

[75]:
```
#predict
#predict
predictions=NAND.predict(X)
k=0
for i in predictions:

    if( i> 0.5):
        print("input: ", X[k],"=" ,1)
    else:

        print("input: ", X[k],"=" ,0)
    k+=1
```

```
input:  [0 0] = 1
input:  [0 1] = 1
input:  [1 0] = 1
input:  [1 1] = 0
```

[ ]:
```
```

[ ]: