Keras 2 - Building a Neural Network in Keras - Quiz

March 31, 2018

In this quiz, we will build a simple multi-layer feedforward neural network to solve the XOR problem.

Recall XOR

```
------------------
A   |  B   | A XOR B
------------------
0      0        0
0      1        1
1      0        1
1      1        0
```

Here's what we are going to do:

1. Set the first layer to a `Dense()` layer with an output width of 8 nodes and the `input_dim` set to the size of the training samples (in this case 2).
2. Add a `tanh` activation function.

3. Set the output layer width to 1, since the output has only two classes. (We can use 0 for one class and 1 for the other).
4. Use a `sigmoid` activation function after the output layer.
5. Run the model for 50 epochs.

This should give you an accuracy of 50%. That's OK, but certainly not great. Out of 4 input points, we are correctly classifying only 2 of them. Let's try to change some parameters around to improve. For example, you can increase the number of epochs. You'll pass this quiz if you get 75% accuracy.

Finished work:

```
In [2]: import numpy as np
        from keras.utils import np_utils
        import tensorflow as tf
        tf.python_io.control_flow_ops = tf

        # Set random seed
        np.random.seed(42)

        # Trained data: X and y
```

```
X = np.array([[0,0],[0,1],[1,0],[1,1]]).astype('float32') # (4,2)
y = np.array([[0],[1],[1],[0]]).astype('float32') # (4,1)

# Initial setup for Keras
from keras.models import Sequential
from keras.layers.core import Dense, Activation

# Building the model
xor = Sequential()

# 1. Set the first layer to a Dense() layer with an output width of 8 nodes
#    and the input_dim set to the size of the training samples (in this case 2).
xor.add(Dense(8, input_dim = X.shape[1]))

# 2. Add a tanh activation function.
xor.add(Activation('tanh'))

# 3. Set the output layer width to 1, since the output has only two classes.
#    (We can use 0 for one class and 1 for the other).
xor.add(Dense(1))

# 4. Use a sigmoid activation function after the output layer.
xor.add(Activation('sigmoid'))

# Specify loss as "binary_crossentropy", optimizer as "adam",
# and add the accuracy metric
xor.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Print the model architecture
xor.summary()

# Fitting the model
#history = xor.fit(X, y, epochs=50, verbose=0)
# Obtained Accuracy:  0.5
history = xor.fit(X, y, epochs=1000, verbose=0)

# Scoring the model
score = xor.evaluate(X, y)
print('\nAccuracy: ', score[-1])

# Checking the predictions
print('\nPredictions:')
print(xor.predict_proba(X))
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 8)                 24
```

```
-----------------------------------------------------------------
activation_3 (Activation)    (None, 8)                 0
-----------------------------------------------------------------
dense_4 (Dense)              (None, 1)                 9
-----------------------------------------------------------------
activation_4 (Activation)    (None, 1)                 0
=================================================================
Total params: 33
Trainable params: 33
Non-trainable params: 0
-----------------------------------------------------------------
4/4 [==============================] - 0s 6ms/step

Accuracy:  1.0

Predictions:
[[0.06305031]
 [0.8665381 ]
 [0.9114123 ]
 [0.11303615]]
```