XOR

February 12, 2021

1 XOR con Keras

1.1 Deep Learning

Brandon Alain Cruz Ruiz

Oscar Allan Ruiz Toledo

Inicializamos nuestras entradas y nuestras salidas que servirán para el modelo.

[ ]:
```
import tensorflow as tf
import numpy as np

inputs = np.array([[0,0],[0,1],[1,0],[1,1]], "float32")
outputs = np.array([[0],[1],[1],[0]], "float32")
```

Utilizaremos keras para la creación del modelo, en este primer ejemplo utilizaremos una red oculta de 16 nodos con la función de activación Relu para la capa oculta y Sigmoid para la capa de salida.

[ ]:
```
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(16, input_dim=2, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')])
```

Pensabamos utilizar funciones de pérdida y optimizadores "externos" como los ejercicios vistos en clase, pero fue más sencillo agregarlos directamente al modelo con las funciones predefinidas de keras. Utilizamos la función de pérdida de entropía cruzada binaria, pues de acuerdo al artículo leído esta función de pérdida es la mas adecuada para este ejemplo, y el optimizador de gradiente descendiente.

[134]:
```
model.compile(loss='binary_crossentropy',
              optimizer='sgd',
              metrics=['binary_accuracy'])
```

[134]:
```
<tensorflow.python.keras.callbacks.History at 0x7f6f88197be0>
```

Finalmente entrenamos el modelo. En este caso fue un poco de prueba y error para determinar el número adecuado de epochs para poder optener una precisión adecuada para el modelo. En este caso utilizamos 480 epochs, pero viendo los resultados, más o menos obtenemos los resultados esperados aproximadamente a partir de la epoch 460.

[148]:
```
model.fit(inputs, outputs, verbose=0, epochs=480)
```

[148]:
```
<tensorflow.python.keras.callbacks.History at 0x7f6f82c9dfd0>
```

Y al momento de realizar una predicción, logramos los resultados esperados.

Nota: Al aumentar las epochs a 500, el último valor que debería ser 0 se volvía 1.

[149]:
```
print(model.predict(inputs).round())
```

```
[[0.]
 [1.]
 [1.]
 [0.]]
```

Realizamos un segundo modelo con una segunda capa oculta igual de 16 nodos para observar si existían mejoras en el número de epochs que se realizan.

[150]:
```
model2 = tf.keras.models.Sequential([
    tf.keras.layers.Dense(16, input_dim=2, activation='relu'),
    tf.keras.layers.Dense(16, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')])

model2.compile(loss='binary_crossentropy',
               optimizer='sgd',
               metrics=['binary_accuracy'])

model2.fit(inputs, outputs, verbose=0, epochs=300)
```

[150]:
```
<tensorflow.python.keras.callbacks.History at 0x7f6f82c00c50>
```

Agregando una segunda capa oculta, observamos mejoría evidente. El número de epochs necesarios para alcanzar una precisión de 1 es aproximadamente a partir de la epoch número 230.

[151]:
```
print(model2.predict(inputs).round())
```

```
WARNING:tensorflow:7 out of the last 11 calls to <function
Model.make_predict_function.<locals>.predict_function at 0x7f6f8c5c60d0>
triggered tf.function retracing. Tracing is expensive and the excessive number
of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2)
passing tensors with different shapes, (3) passing Python objects instead of
tensors. For (1), please define your @tf.function outside of the loop. For (2),
@tf.function has experimental_relax_shapes=True option that relaxes argument
shapes that can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
[[0.]
 [1.]
 [1.]
```

```
[0.]]
```

Estos fueron los sitios que nos ayudaron a crear esta red neuronal utilizando keras:

- Burgdorf, G. (2016). *Understanding XOR with Keras and TensorFlow.* Thoughtram. https://blog.thoughtram.io/machine-learning/2016/11/02/understanding-XOR-with-keras-and-tensorlow.html
- Pal, L. (2019). *Understanding Basics of Deep Learning by solving XOR problem.* Medium. https://medium.com/analytics-vidhya/understanding-basics-of-deep-learning-by-solving-xor-problem-cb3ff6a18a06