**03008915** C/ Ferrocaril, 22, 03570 La Vila Joiosa **Tel** 966870140 **Fax** 966870141 http://portal.edu.gva.es/iesmarcoszaragoza

# 1 Guardar y cargar modelo de Red Neuronal

Dado que los modelos de aprendizaje profundo pueden tardar horas, días e incluso semanas en entrenarse, es importante saber cómo guardarlos y cargarlos desde disco. Vamos a ver la manera de:

- Cómo guardar los pesos del modelo y la arquitectura del modelo en archivos separados
- Cómo guardar la arquitectura del modelo en formato JSON
- Cómo guardar los pesos y la arquitectura del modelo en un único archivo para su uso posterior

## 1.1 Introducción

Keras separa la arquitectura del modelo y los pesos de este.
Los pesos del modelo se guardan en un formato HDF5. Este formato de rejilla es ideal para almacenar matrices multidimensionales de números.
La estructura del modelo puede describirse y guardarse utilizando JSON.

- Guardar modelo en JSON
- Guardar modelo en HDF5
-

El primero guarda la arquitectura y los pesos del modelo **por separado**. Los pesos del modelo se guardan en un archivo con formato HDF5 en todos los casos.

**Notas:**
Se debe tener TensorFlow v2.x instalado.
Guardar modelos requiere tener instalada la librería h5py. Normalmente se instala como una dependencia con TensorFlow.

## 1.2 Guardar el modelo de red neuronal en JSON

JSON es un formato de archivo simple para describir datos jerárquicamente.
Keras proporciona la capacidad de describir cualquier modelo utilizando el formato JSON con una función `to_json()`. Esto puede ser guardado en un archivo y posteriormente cargado a través de la función `model_from_json()` que creará un nuevo modelo a partir de la especificación JSON.

Los pesos se guardan directamente desde el modelo mediante la función `save_weights()` y se cargan posteriormente mediante la función simétrica `load_weights()`.

Este ejemplo entrena y evalúa un modelo sencillo en el "Pima indian dataset". El modelo se convierte a formato JSON y se escribe en **model.json** en el directorio local. Los pesos de la red se escriben en **model.h5** en el directorio local.

**03008915** C/ Ferrocaril, 22, 03570 La Vila Joiosa **Tel** 966870140 **Fax** 966870141 http://portal.edu.gva.es/iesmarcoszaragoza

```
from tensorflow.keras.models import Sequential, model_from_json
from tensorflow.keras.layers import Dense
import numpy
import os

numpy.random.seed(7)
# load dataset
dataset = numpy.loadtxt("pima-indians-diabetes.csv", delimiter=",")

X = dataset[:,0:8]
Y = dataset[:,8]
# crear modelo
model = Sequential()
model.add(Dense(12, input_dim=8, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
# Compilar modelo
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
# Entrenar
model.fit(X, Y, epochs=150, batch_size=10, verbose=0)
# evaluar
scores = model.evaluate(X, Y, verbose=0)
print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))

# Guardar modelo serializado en JSON
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
# Serializar pesos en HDF5
model.save_weights("model.h5")
```

Hasta aquí hemos creado el modelo y guardado en disco.

## 1.3 Carga del modelo desde ficheros

Los datos del modelo y los pesos se cargan desde los archivos en disco y se crea un nuevo modelo. **Es importante compilar el modelo cargado antes de utilizarlo**. Esto es para que las predicciones hechas usando el modelo puedan usar la computación eficiente apropiada del backend Keras.

Para cargar un modelo pre-entrenado, hacemos lo siguiente:

```
# cargar json
json_file = open('model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)

# cargar pesos en el modelo
loaded_model.load_weights("model.h5")

# evaluar modelo
loaded_model.compile(loss='binary_crossentropy', optimizer='rmsprop', metrics=['accuracy'])
score = loaded_model.evaluate(X, Y, verbose=0)
print("%s: %.2f%%" % (loaded_model.metrics_names[1], score[1]*100))
```

**Nota**: Los resultados pueden variar debido a la naturaleza estocástica del algoritmo o del procedimiento de evaluación, o a diferencias en la precisión numérica.

**03008915** C/ Ferrocaril, 22, 03570 La Vila Joiosa **Tel** 966870140 **Fax** 966870141 http://portal.edu.gva.es/iesmarcoszaragoza

## 1.4 Guardar los pesos y la arquitectura del modelo juntos

Keras también soporta una interfaz más simple para guardar los pesos y la arquitectura del modelo juntos en un único archivo H5.

Guardar el modelo de esta manera incluye todo lo que se necesita saber sobre el modelo, incluyendo:

- Pesos del modelo
- Arquitectura del modelo
- Detalles de compilación del modelo (pérdidas y métricas)
- Estado del optimizador del modelo

Esto significa que se puede cargar y utilizar el modelo directamente sin tener que volver a compilarlo como anteriormente.

**Nota**: Esta es la mejor manera de guardar y cargar el modelo Keras.

```
from tensorflow.keras.models import Sequential, model_from_json
from tensorflow.keras.layers import Dense
import numpy
import os

numpy.random.seed(7)
# load dataset
dataset = numpy.loadtxt("pima-indians-diabetes.csv", delimiter=",")

X = dataset[:,0:8]
Y = dataset[:,8]
# crear modelo
model = Sequential()
model.add(Dense(12, input_dim=8, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
# Compilar modelo
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
# Entrenar
model.fit(X, Y, epochs=150, batch_size=10, verbose=0)
# evaluar
scores = model.evaluate(X, Y, verbose=0)
print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))

# Guardar modelo y arquitectura en un fichero
model.save("model.h5")
```

o bien, una manera equivalente con una función específica:

```
from tensorflow.keras.models import save_model
save_model(model, "model.h5")
```

**03008915** C/ Ferrocaril, 22, 03570 La Vila Joiosa **Tel** 966870140 **Fax** 966870141 http://portal.edu.gva.es/iesmarcoszaragoza

## 1.5 Cómo cargar un modelo Keras

El modelo guardado puede ser cargado más tarde llamando a la función `load_model()` y pasándole el nombre del archivo. La función devuelve el modelo con la misma arquitectura y pesos.

En este caso, se carga el modelo, la arquitectura y se evalúa en el mismo conjunto de datos para confirmar que los pesos y la arquitectura son los mismos.

```
from numpy import loadtxt
from tensorflow.keras.models import load_model

# cargar modelo
model = load_model('model.h5')
# hacer un resumen para ver si es el mismo.
model.summary()
# carga de csv
dataset = loadtxt("pima-indians-diabetes.csv", delimiter=",")
# Features/ target
X = dataset[:,0:8]
Y = dataset[:,8]
# Evaluar
score = model.evaluate(X, Y, verbose=0)
print("%s: %.2f%%" % (model.metrics_names[1], score[1]*100))
```