Universidad de los Andes Colombia | Facultad de Ingeniería

Universidad de los Andes
Maestría en Inteligencia Analítica para la Toma de Decisiones.
Modelos Avanzados para el Análisis De Datos 2.
Prof. Martín Andrade Restrepo.

Taller 1. Introducción práctica a Redes Neuronales. Introducción a Keras y Tensor Flow

Para los siguientes dos ejercicios:

Deberá realizarlos sin la implementación de código en Python.

Ejercicio I

(10 pts) Explique muy brevemente y con sus propias palabras los siguientes conceptos (puede investigarlos pero al escribir trate de no tener ninguna fuente abierta).

a) Inteligencia artificial.

b) Machine learning (aprendizaje de máquinas).

c) Aprendizaje supervisado, no supervisado y reforzado.

d) Datos de entrenamiento, prueba y validación.

e) Método de validación cruzada.

Ejercicio II

(10 pts) Diseñe una red de perceptrones usando el modelo McCulloch-Pitts (cada neurona tiene una entrada y una salida binaria) que sea equivalente a las funciones lógicas "or" y "and" para una entrada binaria.

Para los siguientes dos ejercicios:

Deberá realizarlos con la implementación de código en Python.

Ejercicio III

(40 pts) Creación y entrenamiento de una red neuronal.

Recomendaciones:

- Para este taller se podrá excepcionalmente trabajar con Google Colab. Para utilizar Colab, haga clic aquí.
- Para los siguientes ya no será posible. De igual manera, se recomienda resolver el taller usando Spyder y/o en Pycharm (son dos de los entornos de desarrollo para Python más utilizados por Data Scientists y Data Analysts).
- Además, se recomienda crear un "Virtual Environment" (Venv) de Python para trabajar en los talleres del curso. En dicho ambiente, se instalarán las librerías (en sus versiones adecuadas) a ser utilizadas en los talleres y el proyecto.
- Para descargar Anaconda, haga clic aquí.
- Para descargar PyCharm, haga clic aquí.
- Para descargar Spyder, haga.

Pasos a seguir (si utiliza Google Colab, saltar Pasos 1-6):

1. Una vez descargado Anaconda y Spyder/PyCharm, instale Anaconda.
2. Cuando Anaconda haya sido instalado, ejecute la aplicación ***Anaconda Prompt*** (como administrador). Anaconda Prompt es una ventana de terminal. En ella crearemos un "environment" de python (los environments sirven para aislar proyectos con sus determinados requerimientos y versiones de librerías). Seleccione el directorio donde guardará en "environment".
3. - Si su computador cuenta con gpu (que permita el uso de CUDA y/o computación en paralelo) puede ejecutar el comando:

```
conda create --name tf tensorflow-gpu
```

   (las librerías usadas en TF se instalarán automáticamente)

   - Si su computador no cuenta con GPU sino con una CPU normal, puede ejecutar (línea por línea):

```
conda create --name tf_cpu
conda activate tf_cpu
conda install pip
pip install tensorflow
conda deactivate
```

   Estás líneas crean el "environment", lo activan, instalan la herramienta de instalación `pip` que luego se usa para instalar TF.

4. Una vez creado el environment, instale PyCharm o Spyder (como administrador). Una vez instalado, ejecute el programa.

5. Si utiliza PyCharm:

   - Cree un nuevo proyecto.
   - Seleccione el directorio y el nombre donde será guardado.
   - Al crearlo, seleccione las opciones `Existing interpreter` → `Conda environment` → `Existing environment`.
   - Debería aparecerle el environment que creó previamente (ej. tf_cpu). Selecciónelo y al final haga clic en `Create project`.

6. Si utiliza Spyder:

   - Haga clic en `Herramientas` → `Preferencias`.
   - En `Intérprete` seleccione el "enviroment" que creó.
   - Al realizar los pasos anteriores debería abrirse automáticamente un scrypt de código que puede llamar main.py (el código "principal" de su proyecto).

7. Verificaremos que estemos usando TF2 corriendo el main con las líneas:

```
import tensorflow as tf
print(tf.__version__)
```

   Al correr estas líneas debería aparecer en el prompt una versión de TF superior a la 2.0.

   - Si usted está utilizando su GPU, añada las siguientes líneas después de la importación de TF (para evitar errores de computación):

```
import tensorflow as tf
physical_devises = tf.config.list_physical_devices('GPU')
tf.config.experimental.set_memory_growth(physical_devices[0],True)
```

En este taller programaremos nuestra primer red neuronal. Utilizaremos dicha red para reconocer dígitos. Usaremos el conjunto de datos MNIST, que cuenta con miles de imágenes de dígitos escritos a mano de 28×28 pixeles en escala de grises. El conjunto cuenta con un conjunto de entrenamiento de 60 mil imágenes y un conjunto de prueba de 10 mil imágenes (tomadas de forma independiente).

Investigue sobre el conjunto MNIST. Describa sus observaciones.

Para cargar los datos, utilice:

```
from tensorflow.keras.datasets import mnist
from tensorflow import keras
from tensorflow.keras import layers
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

Verifique las dimensiones de los datos:

```
train_images.shape
len(train_labels)
train_labels
test_images.shape
len(test_labels)
test_labels
```

Describa sus observaciones sobre los datos. Puede realizar también los análisis que usted considere pertinentes (distribuciones etc.)

Para que nuestra red reciba un vector y no una matriz, necesitamos cambiar la dimensionalidad de nuestros datos. Para eso podemos usar:

```
train_images = train_images.reshape((60000, 28 * 28))
train_images = train_images.astype("float32") / 255
test_images = test_images.reshape((10000, 28 * 28))
test_images = test_images.astype("float32") / 255
```

Describa lo que hacen estas líneas de código.

Cree una red neuronal usando neuronas sigmoides y una capa final softmax. Para esto puede usar la función `Sequential` de `keras`:

```
model = keras.Sequential([
    layers.Dense(512, activation="sigmoid"),
    layers.Dense(10, activation="softmax")
])
```

Investigue y describa lo que hace la función `sequential` y lo que es la activación `softmax`.

Para compilar su modelo (su red) use la función `compile`:

```
model.compile(optimizer="SGD",
loss="SparseCategoricalCrossentropy",
metrics=["accuracy"])
```

Investigue y describa lo que hace la función `compile`:

Luego, puede entrenar su modelo con la función `fit`:

```
model.fit(train_images, train_labels, epochs=5, batch_size=128)
```

Investigue y describa lo que hace la función `fit`. Describa sus resultados al correr dichas líneas.

Para evaluar su modelo puede usar:

```
test_loss, test_acc = model.evaluate(test_images, test_labels)
```

Investigue y describa lo que hace la función `evaluate`.

Ejercicio IV

(40 pts) Utilizando lo aprendido en el numeral anterior, diseñe por lo menos 4 redes diferentes (con distinto número de capas intermedias, distintos números de neuronas en ellas y distintas activaciones). Inicie y entrene sus redes múltiples veces. Describa las redes usadas y sus resultados.

¿Cuál es el mejor desempeño que logra conseguir (porcentaje de acierto con el conjunto de prueba)?