IMPLEMENTADO MACHINE LEARNING
Práctica 5 – Deep Learning

Material de lectura:

- Francois Chollet. Deep Learning with Python. Capítulo 5.
- Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning. Capítulos 9, 11 y 12.

1. **MNIST.** Es una base de datos que contiene 70 mil imágenes en escala de grises de dígitos escritos a mano. Genere un modelo que permita clasificar las 10 clases.

   a) Carga de datos

   El código para descargar el dataset ya se encuentra en Keras y es posible cargarlo en memoria del siguiente modo:

   from keras.datasets import mnist

   (X_train, Y_train), (X_test, Y_test) = mnist.load_data()

   Los arreglos X_train y X_test poseen las imágenes del dataset y cada uno tiene un shape: (N, H,W), donde N es la cantidad de ejemplos, H el alto de la imagen y W el ancho. En el caso de MNIST tanto H como W valen 28.

   b) Visualización

   El primer paso para cerciorarse de que un conjunto de datos de imágenes está cargado correctamente es visualizar las mismas.

   Para visualizar las imágenes, puede utilizar la función plt.imshow(). Por ejemplo, para ver la primera imagen del conjunto de entrenamiento, puede ejecutar:

   plt.imshow(X_train[0, :,:])

   Para ver la tercera imagen del conjunto de prueba:

   plt.imshow(X_test[2, :,:])

   Implemente un código que visualice 8 imágenes de cada clase en una grilla de 8x10. Utiliza la función *subplots* de *matplotlib.pyplot* para generar la grilla.

   c) Aplanamiento

   Para clasificar con una red neuronal con capas del tipo x*W+b (clase *Dense* en Keras), el formato de imagen (N,H,W) no sirve. Por ende los ejemplos deben adaptarse al formato (N,V), donde V es la cantidad de variables, que en este caso sería V=H*W=28*28= 784. La capa *Flatten* realiza esta operación por nosotros:

   model.add(Flatten(input_shape=(28,28)))

   Nota: La API de Keras es inconsistente, y utiliza el parámetro input_shape en la clase Flatten, mientras que en el resto de capas se utiliza el parámetro input_dim. Recordamos que este parámetro sólo es necesario en la primera capa para especificar el tamaño de la entrada.

   d) Entrenamiento del modelo

Entrene un modelo para clasificar las imágenes, utilizando una softmax en la capa de salida y la entropía cruzada como función de error. Mida el error y el accuracy en el conjunto de test (y el de train). Compute la matriz de confusión, pero antes de mirarla ¿qué pares de clases le parece que van a confundirse más?

e) Normalización de las imágenes

Los pixeles de las imágenes están codificados en el rango 0-255. Es más beneficioso para el entrenamiento de la red que estén normalizadas con media 0 y varianza 1. La normalización debe realizarse a nivel de pixel, es decir, se debe calcular la media de todos los pixeles de todas las imágenes, luego la varianza de todos los pixeles de todas las imágenes y luego realizar la normalización. Esto lo puede realizar con Sk-learn como en las prácticas anteriores.

f) CNN en MNIST

Diseñe un modelo de Redes Neuronales Convolucionales (CNN) para clasificar el conjunto de datos MNIST. Pruebe varios modelos distintos, variando los siguientes hiperparámetros (Recuerde evaluar en el conjunto de testing):

- Cantidad de capas Conv
- Cantidad de feature maps (kernels)
- Stride, padding y kernel_size de las convoluciones
- Funciones de activación (ej: ReLU)
- Uso de capas Max Pooling.
- Cantidad de neuronas ocultas en la capa Dense.

2. CIFAR10. Este conjunto de datos posee imágenes de 10 clases de la vida cotidiana como diferentes animales y vehículos.

a) Carga de datos

```
from keras.datasets import cifar10

(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

b) Visualización

Para visualizar las imágenes también puede utilizar la función plt.imshow(), pero ahora debe tener en cuenta la dimensión extra:

```
plt.imshow(x_train[0,:,:,:]) #se incluye la dimensión de canales
```

c) Aplanamiento

Las imágenes de CIFAR10 son a color, por ende las dimensiones de x_train y x_test son del tipo (N,H,W,C), donde C es la cantidad de canales (3 para imágenes a color). En el caso de CIFAR10, las imágenes son de tamaño 32x32, por ende el *shape* de estos vectores es (N,32,32,3). En este caso también tenemos que utilizar la capa Flatten al principio del modelo denso, pero ahora con el nuevo tamaño de entrada:

```
model.add(Flatten(input_shape=(32,32,3)))
```

d) Entrenamiento del modelo

Entrene un modelo para clasificar las imágenes, utilizando una softmax en la capa de salida y la entropía cruzada como función de error. Mida el error y el accuracy en el conjunto de test (y el de

train). Compute la matriz de confusión, pero antes de mirarla ¿qué pares de clases te parece que van a confundirse más? Recuerde normalizar los datos con anterioridad.

e) CNN en CIFAR10

Repita el mismo procedimiento que para MNIST para el conjunto de datos CIFAR10. El entrenamiento para CIFAR10 generalmente tarda más tiempo que para MNIST debido a que las imágenes tienen aproximadamente tres veces la dimensionalidad y además mayor complejidad. Por ende, pruebe sólo algunas arquitecturas distintas. Si tiene instalado Keras/TensorFlow para GPU, el entrenamiento puede ser mucho más rápido.