Funciones dadas para los apartados 1 y 2 de la práctica 2: Redes Neuronales Convolucionales

Visión por Computador

Anabel Gómez Ríos
anabelgrios@decsai.ugr.es

Universidad de Granada

25 de octubre de 2019

Índice

Índice

Cargar las librerías necesarias (1/2)

```
# En caso de necesitar instalar keras en google colab,
# ejecutar la siguiente línea:
# !pip install -q keras
# Importar librerías necesarias
import numpy as np
import keras
import matplotlib.pyplot as plt
import keras.utils as np_utils

# Importar modelos y capas que se van a usar
# A completar
```

Cargar las librerías necesarias (2/2)

```
# Importar el optimizador a usar
from keras.optimizers import SGD

# Importar el conjunto de datos
from keras.datasets import cifar100
```

Índice

Lectura y modificación del conjunto de imágenes (1/3)

```
# A esta función sólo se le llama una vez. Devuelve 4
# vectores conteniendo, por este orden, las imágenes
# de entrenamiento, las clases de las imagenes de
# entrenamiento, las imágenes del conjunto de test y
# las clases del conjunto de test.

def cargarImagenes():
  # Cargamos Cifar100. Cada imagen tiene tamaño
  # (32, 32, 3). Nos vamos a quedar con las
  # imágenes de 25 de las clases.

  (x_train, y_train), (x_test, y_test) =
            = cifar100.load_data(label_mode='fine')
  x_train = x_train.astype('float32')
  x_test = x_test.astype('float32')
```

Lectura y modificación del conjunto de imágenes (2/3)

```
x_train /= 255
x_test /= 255

train_idx = np.isin(y_train, np.arange(25))
train_idx = np.reshape(train_idx,-1)
x_train = x_train[train_idx]
y_train = y_train[train_idx]

test_idx = np.isin(y_test, np.arange(25))
test_idx = np.reshape(test_idx, -1)
x_test = x_test[test_idx]
y_test = y_test[test_idx]
```

Lectura y modificación del conjunto de imágenes (3/3)

```
# Transformamos los vectores de clases en matrices.
# Cada componente se convierte en un vector de ceros
# con un uno en la componente correspondiente a la
# clase a la que pertenece la imagen. Este paso es
# necesario para la clasificación multiclase en keras.

y_train = np_utils.to_categorical(y_train, 25)
y_test = np_utils.to_categorical(y_test, 25)

return x_train, y_train, x_test, y_test
```

Índice

Obtener el accuracy en el conjunto de test

```
# Esta función devuelve el accuracy de un modelo, defi-
# nido como el porcentaje de etiquetas bien predichas
# frente al total de etiquetas. Como parámetros es
# necesario pasarle el vector de etiquetas verdaderas
# y el vector de etiquetas predichas, en el formato de
# keras (matrices donde cada etiqueta ocupa una fila,
# con un 1 en la posición de la clase a la que pertenece
# 0 en las demás).

def calcularAccuracy(labels, preds):
  labels = np.argmax(labels, axis = 1)
  preds = np.argmax(preds, axis = 1)

  accuracy = sum(labels == preds)/len(labels)

  return accuracy
```

Índice

Gráficas de evolución durante el entrenamiento (1/2)

```
# Esta función pinta dos gráficas, una con la evolución
# de la función de pérdida en el conjunto de train y
# en el de validación, y otra con la evolución del
# accuracy en el conjunto de train y el de validación.
# Es necesario pasarle como parámetro el historial del
# entrenamiento del modelo (lo que devuelven las
# funciones fit() y fit_generator()).

def mostrarEvolucion(hist):

  loss = hist.history['loss']
  val_loss = hist.history['val_loss']
  plt.plot(loss)
  plt.plot(val_loss)
  plt.legend(['Training loss', 'Validation loss'])
  plt.show()
```

Gráficas de evolución durante el entrenamiento (2/2)

```
acc = hist.history['acc']
val_acc = hist.history['val_acc']
plt.plot(acc)
plt.plot(val_acc)
plt.legend(['Training accuracy',
                    'Validation accuracy'])
plt.show()
```