TP09-MNIST-AE

September 18, 2019

1 Auto-Encoder on MNIST

We will use the same MNIST dataset with each example shaped as `(28,28,1)` array.

[1]:
```
import numpy as np
import tensorflow as tf
from tensorflow import keras

import matplotlib.pyplot as plt
%matplotlib inline
import datetime as dt

from pathlib import Path

from sklearn.datasets import fetch_openml
from sklearn import preprocessing

import io
import scipy

data_home = '/tmp/scikit_learn_data/'
datafile = '/tmp/mnist.npz'

datapath = Path(datafile)
if not(datapath.exists()):
    print("Data File not found... downloading it")
    Xmnist, ymnist = fetch_openml('mnist_784',
                                  version=1,
                                  return_X_y=True,
                                  data_home=data_home)
    np.savez(datapath.as_posix(),
             X=np.array(Xmnist, dtype='u8'),
             y=np.array(ymnist, dtype='u8'))
    print("Data File downloaded and saved")
    del Xmnist, ymnist

print("Data File found... loading it into memory")
data = np.load(datapath.as_posix())
```

```
Xmnist = data['X']/255.
ymnist = keras.utils.to_categorical(data['y'])
print("Data File loaded")

Xtrain, Ytrain = Xmnist[:50000], ymnist[:50000]
Xvalid, Yvalid = Xmnist[50000:60000], ymnist[50000:60000]
Xtest, Ytest = Xmnist[-10000:], ymnist[-10000:]

Xtrain = Xtrain.reshape((Xtrain.shape[0], 28, 28, 1))
Xvalid = Xvalid.reshape((Xvalid.shape[0], 28, 28, 1))
Xtest = Xtest.reshape((Xtest.shape[0], 28, 28, 1))


def ndarray2image_summary(array, name='Image'):
    # Convert a ndarray in float32 to an 8bits grayscale summary
    intarray = np.array(array*255, dtype='u8')
    height, width = intarray.shape
    with io.BytesIO() as output:
        scipy.misc.imsave(output, intarray, format='png')
        image_string = output.getvalue()

    imageSummary = tf.Summary.Image(height=height,
                                    width=width,
                                    colorspace=1,
                                    encoded_image_string=image_string)
    summary = tf.Summary(
        value=[tf.Summary.Value(tag=name, image=imageSummary)])

    return summary


class CustomTensorBoard(keras.callbacks.TensorBoard):
    # CustomTensorBoard callback that logs prediction on image_set into images.
    def __init__(self, *args, **kwargs):
        self._input_image_set = kwargs.pop('input_image_set', None)
        super().__init__(*args, **kwargs)

    def on_train_begin(self, logs=None):
        super().on_train_begin(logs)
        if self._input_image_set is not None:
            for i in range(self._input_image_set.shape[0]):
                summary = ndarray2image_summary(
                    self._input_image_set[i, :, :, 0],
                    name="Input image %02d" % (i,),)
                self.writer.add_summary(summary,0)

    def on_epoch_end(self, epoch, logs=None):
```

```
        super().on_epoch_end(epoch, logs)
        if self._input_image_set is not None:
            predicted_images = self.model.predict(self._input_image_set)
            for i in range(predicted_images.shape[0]):
                summary = ndarray2image_summary(
                    predicted_images[i, :, :, 0],
                    name="Predicted image %02d" % (i,),)
                self.writer.add_summary(summary, epoch)
```

```
Data File found... loading it into memory
Data File loaded
```

1.1 Output images to Tensorboard

If a model outputs an image (that's the case this auto-encoder), you can log images to Tensorboard by using a custom Tensorboard callback.

Observe the following code `ndarray2image_summary` and `ndarray2image_summary` above, the custom class can be instantiated like this:

```
tensorboard_callback = CustomTensorBoard(log_dir=logdir,
                                         write_grads=True,
                                         histogram_freq=1,
                                         input_image_set=Xvalid[:10])
```

Here a custom tensorboard callback has been created with the 10 first images of the valid set as input. At each epoch end, this images will be passed to the auto-encoder and log to tensorboard.

1.2 Exercises

1) Build an auto-encoder with only dense layers.

2) Build an auto-encoder with convolutional layers followed by dense layers for the Encoder and with dense layer followed by transpose convolutional layers for the decoder.

3) Add skip connection to dense layers in the middle of the network