Q2_CAE

March 24, 2022

1 Allowing Import from Parent Directory

[2]:
```
import os
import sys
import inspect

currentdir = os.path.dirname(os.path.abspath(inspect.getfile(inspect.
 ↪currentframe())))
parentdir = os.path.dirname(currentdir)
sys.path.insert(0, parentdir)
```

2 Importing Packages

[4]:
```
import glob
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import time
import tools.loaddata as loaddata
import tools.dataassimilation as da
import tools.visualisation as visual

import sklearn
assert sklearn.__version__ >= "0.20"
from sklearn.metrics import mean_squared_error
from sklearn.decomposition import PCA

# TensorFlow 2.0 is required
import tensorflow as tf
from tensorflow import keras
assert tf.__version__ >= "2.0"
```

3 Loading and Reshaping Data

[ ]:
```
path_train = "../data/train/"
path_test = "../data/test/"
path_back = "../data/background/"
path_obs = "../data/satellite/"
```

[6]:
```
train_full = loaddata.load_data(path_train)[:,:858,:910]
test = loaddata.load_data(path_test)[:,:858,:910]
model_data = loaddata.load_data(path_back)[:,:858,:910]
satellite_data = loaddata.load_data(path_obs)[:,:858,:910]
```

[72]:
```
# Due to RAM and time limitations when running the convolutional autoencoder,␣
 ↪we had to reduce the dataset massively.
```

[7]:
```
train = train_full[0:100]
```

[8]:
```
print(f"Train data shape reshaping: {np.shape(train)}")
print(f"Test data shape reshaping: {np.shape(test)}")
print(f"Background data shape reshaping: {np.shape(model_data)}")
print(f"Observational data shape reshaping: {np.shape(satellite_data)}")
```

```
Train data shape reshaping: (100, 858, 910)
Test data shape reshaping: (300, 858, 910)
Background data shape reshaping: (5, 858, 910)
Observational data shape reshaping: (5, 858, 910)
```

4 Convolutional Autoencoder

[9]:
```
np.random.seed(42)
tf.random.set_seed(42)

encoder = keras.models.Sequential([keras.layers.
 ↪Conv2D(16,(9,9),padding='same',activation='relu'),
                                    keras.layers.MaxPooling2D((13,13)),
                                    keras.layers.
 ↪Conv2D(1,(7,7),padding='same',activation='relu'),
                                    keras.layers.MaxPooling2D((11, 7)),
                                    keras.layers.Flatten(),
                                    ])
decoder = keras.models.Sequential([
                                   keras.layers.Reshape((6,10,1)),
                                   keras.layers.Conv2D(1, (5, 5),␣
 ↪padding='same', activation='relu'),
                                   keras.layers.UpSampling2D((11, 7)),
```

```
                                    keras.layers.Conv2D(16, (7, 7),␣
↪padding='same', activation='relu'),
                                    keras.layers.UpSampling2D((13, 13)),
                                    keras.layers.Conv2D(1, (9, 9),␣
↪padding='same', activation='sigmoid'),
                                    ])
autoencoder = keras.models.Sequential([encoder, decoder])

autoencoder.compile(loss='binary_crossentropy',
                    optimizer=keras.optimizers.Adam(),
                    metrics=['mse'])
print('Encoder:')
encoder.build(input_shape=(None,858,910,1))
encoder.summary()
print('\nDencoder:')
decoder.build(input_shape=(None,60,1))
decoder.summary()
print('\nAutoencoder:')
autoencoder.build(input_shape=(None,858,910,1))
autoencoder.summary()
```

```
Encoder:
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 858, 910, 16)      1312

 max_pooling2d (MaxPooling2D  (None, 66, 70, 16)       0
 )

 conv2d_1 (Conv2D)           (None, 66, 70, 1)         785

 max_pooling2d_1 (MaxPooling  (None, 6, 10, 1)         0
 2D)

 flatten (Flatten)           (None, 60)                0

=================================================================
Total params: 2,097
Trainable params: 2,097
Non-trainable params: 0
_________________________________________________________________

Dencoder:
Model: "sequential_1"
_________________________________________________________________
```

```
 Layer (type)                 Output Shape              Param #
=================================================================
 reshape (Reshape)            (None, 6, 10, 1)          0

 conv2d_2 (Conv2D)            (None, 6, 10, 1)          26

 up_sampling2d (UpSampling2D  (None, 66, 70, 1)         0
 )

 conv2d_3 (Conv2D)            (None, 66, 70, 16)        800

 up_sampling2d_1 (UpSampling  (None, 858, 910, 16)      0
 2D)

 conv2d_4 (Conv2D)            (None, 858, 910, 1)       1297

=================================================================
Total params: 2,123
Trainable params: 2,123
Non-trainable params: 0
_________________________________________________________________

Autoencoder:
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 sequential (Sequential)      (None, 60)                2097

 sequential_1 (Sequential)    (None, 858, 910, 1)       2123

=================================================================
Total params: 4,220
Trainable params: 4,220
Non-trainable params: 0
_________________________________________________________________
```

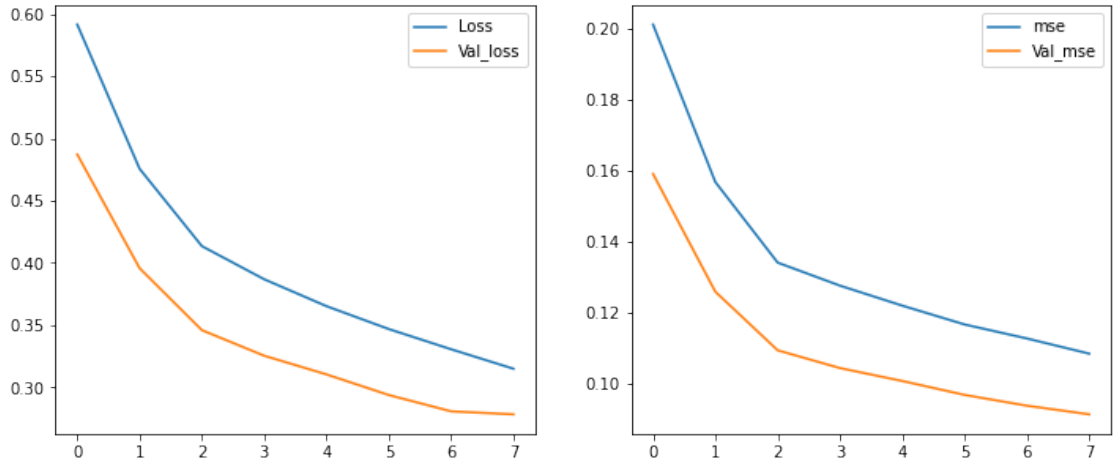
[10]:
```
# We use the following callback to prevent overfitting

from keras import callbacks
earlystopping = callbacks.EarlyStopping(monitor ="val_loss",
                                        mode ="min", patience = 5,
                                        restore_best_weights = True)
```

[ ]:
```
# Due to RAM limitations we were unable to run a bigger batch size. Also, due
# to time we could only run a small number of epochs.
```

[ ]:
```
start = time.time()
history = autoencoder.fit(train,
                          train,
                          epochs=8,
                          batch_size=8,
                          verbose=2,
                          validation_data = (test, test),
                          shuffle = True, callbacks = [earlystopping])
time_ae = time.time() - start
print('Execution time: ', time_ae)
```

[12]:
```
fig, axes = plt.subplots(1,2, figsize=(12,5))
axes[0].plot(history.history['loss'])
axes[0].plot(history.history['val_loss'])
axes[0].legend(['Loss', 'Val_loss'])
axes[1].plot(history.history['mse'])
axes[1].plot(history.history['val_mse'])
axes[1].legend(['mse', 'Val_mse'])
```

[12]: <matplotlib.legend.Legend at 0x7f54b5b944d0>



[13]:
```
test_recovered = autoencoder.predict(test)
mse_test = da.mse(test.reshape((10,-1)), test_recovered.reshape((10,-1)))
print('mse: ', mse_test)
```

```
mse:  0.09133880579540682
```

5 Data Assimilation - Kalman Filter (BLUE)

[68]:
```
# Through experimenting with different computations of R, we found the␣
 ↪following to give us the
# lowest MSE after data assimilation.
```

[65]:
```
## Setting the required variables for data assimilation

model_data_compr = encoder.predict(model_data) # Compressing the model Data
satellite_data_compr = encoder.predict(satellite_data) # Compressing the␣
 ↪satellite Data

latent_space = satellite_data_compr.shape[1]
nNodes = latent_space  #latent_space is the size of the compressed variables or␣
 ↪number of principal components used
I = np.identity(nNodes)
R = da.covariance_diagonal_only(satellite_data_compr.T, latent_space) * 500
H = I
B = I*0.000001
```

[66]:
```
## Performing data assimilation
updated_data_array = da.assimilate(B, H, R, model_data_compr,␣
 ↪satellite_data_compr)
```

[67]:
```
## Printing MSE in latent space
mse_before_DA = da.mse(satellite_data_compr, model_data_compr)
mse_after_DA = da.mse(satellite_data_compr, updated_data_array)
print('MSE before assimilation in latent space: ', mse_before_DA )
print('MSE after assimilation in latent space: ', mse_after_DA)

## Printing MSE in Physical space space
updated_data_recon = decoder.predict(updated_data_array)
mse_before_DA_physical = da.mse(satellite_data, model_data)
mse_after_DA_physical = da.mse(satellite_data, updated_data_recon.squeeze())

print('MSE before assimilation in physical space: ', mse_before_DA_physical)
print('MSE after assimilation in physical space: ', mse_after_DA_physical)
```

```
MSE before assimilation in latent space:  0.3622854
MSE after assimilation in latent space:  0.25463773148562235
MSE before assimilation in physical space:  0.12137401060477984
MSE after assimilation in physical space:  0.10920173230580799
```

[ ]:
```
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('Q2_CAE.ipynb')
```

```
--2022-03-24 18:32:09--  https://raw.githubusercontent.com/brpy/colab-
pdf/master/colab_pdf.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)…
185.199.111.133, 185.199.108.133, 185.199.110.133, …
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.111.133|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 1864 (1.8K) [text/plain]
Saving to: ‘colab_pdf.py’

colab_pdf.py        100%[===================>]   1.82K  --.-KB/s    in 0s

2022-03-24 18:32:09 (14.1 MB/s) - ‘colab_pdf.py’ saved [1864/1864]

Mounted at /content/drive/

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Extracting templates from packages: 100%
```