Inverse Problems Exercises: 2024s s02 (non-sc)

https://www.umm.uni-heidelberg.de/miism/

Notes

- Please **DO NOT** change the name of the `.ipynb` file.
- Please **DO NOT** import extra packages to solve the tasks.
- Please put the `.ipynb` file directly into the `.zip` archive without any intermediate folder.

Please provide your personal information

- full name (Name):

YOUR ANSWER HERE

I06: Convolution theorem

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt

from scipy.linalg import toeplitz
```

In [ ]:

```
from urllib.request import urlopen
import matplotlib.image as mpimg

# create a file-like object from the url
file_input = urlopen('https://upload.wikimedia.org/wikipedia/commons/thumb/1/12/

# load the input image
image_input = mpimg.imread(file_input, 'jpg')

# pick the central line as signal f
f_true = image_input[image_input.shape[0] // 2, :]
```

Imaging model

The imaging model can be represented by

$$g = h \otimes f_{\text{true}} = A f_{\text{true}} = \mathcal{F}^{-1}\{\mathcal{F}\{h\}\mathcal{F}\{f_{\text{true}}\}\}.$$

- f_{true} is the input signal
- h is the point spread function (kernel)
- $\otimes$ is the convolution operator
- A is the Toeplitz matrix of h
- $\mathcal{F}$ and $\mathcal{F}^{-1}$ are the Fourier transform operator and inverse Fourier transform operator
- g is the output signal

Gaussian kernel

Implement the Gaussian kernel function h

- Given the standard deviation of the Gaussian σ_h
- Given the kernel size s_h
- Define the origin of the kernels in the middle of the array
- Normalize the kernel, i.e. the sum of the kernel elements equals to 1
- Implement the function `get_gaussian_1d()` (using `numpy.array`)

Generate the Gaussian kernels

- Parameter options of (σ_h, s_h)
  - (1, 5)
  - (4, 21)
  - (7, 35)
  - (20, 35)
- Save the outputs in the variable `list_h_psf` (as `list` of `numpy.array`)

Display the result

- Plot the kernels in `list_h_psf` in the same order of the parameter options in the axes `ax`
- Show the legend in the axes `ax`

In [ ]:

```python
def get_gaussian_1d(sigma, kernel_size):
    """ Returns a gaussian kernel, with a specified kernel size.
    Low pass (blurring) kernel.

    :param sigma: Standard deviation of the Gaussian function.
    :param kernel_size: Kernel size.
    :returns: normalized Gaussian kernel.
    """
# YOUR CODE HERE
raise NotImplementedError()

fig, ax = plt.subplots()  # Create a figure and an axes.
ax.set_title('Gaussian kernel')

# YOUR CODE HERE
raise NotImplementedError()
```

In [ ]:

```python
# This cell contains hidden tests.
```

In [ ]:

```python
# This cell contains hidden tests.
```

Convolution operation

Convolution with the Gaussian kernels $g = h \otimes f_{\text{true}}$ (using `numpy.convolve()`)

- Apply the kernels in `list_h_psf` to `f_true`
- Return the outputs with the same length as `f_true`
- Save the outputs in the variable `list_g_cov` (as `list` of `numpy.array`)

Display the result

- Plot the outputs in `list_g_cov` in the same order of the parameter options in the axes `ax`
- Plot `f_true` in the axes `ax` (after `list_g_cov`)
- Show the legend in the axes `ax`

In [ ]:

```python
fig, ax = plt.subplots(1, 1, figsize = (15, 3))  # Create a figure and an axes.
ax.set_title('Convolution output')

# YOUR CODE HERE
raise NotImplementedError()
```

In [ ]:

```python
# This cell contains hidden tests.
```

Toeplitz matrix

See: https://en.wikipedia.org/wiki/Toeplitz_matrix#Discrete_convolution

Implement the Toeplitz matrix A corresponding to h (using `scipy.linalg.toeplitz()` optionally)

- Given h
- Given signal size s_f
- Take the zero-padding option, i.e the input array values outside the bounds of the array are assigned 0
- Implement the function `get_convolution_matrix()` (using `numpy.array`)

Generate the Toeplitz matrices

- Return the outputs of each kernel in `list_h_psf` for `f_true`
- Save the outputs in the variable `list_A_psf` (as `list` of `numpy.array`)

Display the result

- Plot the matrices in `list_A_psf` as grayscale images in the same order of the parameter options in the subplots of `axs`
- Add proper titles to the subplots of `axs`

In [ ]:
```
def get_convolution_matrix(kernel, n):
    """ Create a Toeplitz matrix for discrete 1d convolution.

    :param kernel: 1d convolution kernel.
    :param n: Size of the signal, which should be convolved with the kernel.
    :returns: 2d matrix of size (n,n) for convolution by matrix-vector multiplic
    """
# YOUR CODE HERE
raise NotImplementedError()

fig, axs = plt.subplots(2, 2, figsize=(10, 10))
fig.suptitle('Toeplitz matrix')

# YOUR CODE HERE
raise NotImplementedError()
```

In [ ]:
```
# This cell contains hidden tests.
```

In [ ]:
```
# This cell contains hidden tests.
```

Convolution with the Toeplitz matrix

Convolution with the Toeplitz matrix $g = Af_{\mathrm{true}}$

- Apply the Toeplitz matrix in `list_A_psf` to `f_true`
- Save the outputs in the variable `list_g_toe` (as `list` of `numpy.array`)

Display the result

- Plot the outputs in `list_g_toe` in the same order of the parameter options in the axes `ax`
- Plot `f_true` in the axes `ax` (after `list_g_toe`)
- Show the legend in the axes `ax`

In [ ]:
```
fig, ax = plt.subplots(1, 1, figsize = (15, 3))  # Create a figure and an axes.
ax.set_title('Convolution with the Toeplitz matrix')

# YOUR CODE HERE
raise NotImplementedError()
```

In [ ]:
```
# This cell contains hidden tests.
```

In [ ]:
```
# This cell contains tests.

for g_cov, g_toe in zip(list_g_cov, list_g_toe):
    np.testing.assert_almost_equal(g_cov, g_toe) # zero-padding
```

Question: Matrix expression

- What's the advantage to use the matrix expression of convolution $g = Af_{\mathrm{true}}$ for inverse problems?

YOUR ANSWER HERE

Fourier transform

Fourier transform of the Gaussian kernels $\mathcal{F}\{h\}$ (using `numpy.fft.fft()`)

- Pad zeros to both sides of the kernels in `list_h_psf`
- Adjust the kernels as long as `f_true`
- Shift the origin of the kernels to the first element of the array
- Apply the Fourier transform to the shifted padded kernels
- Save the outputs in the variable `list_h_fft` (as `list` of `numpy.array`)

Display the result

- Plot the absolute value of the outputs in `list_h_fft` in the same order of the parameter options in the axes `ax`
- Plot the outputs properly in the frequency domain
- Show the legend in the axes `ax`

In [ ]:
```
fig, ax = plt.subplots(1, 1, figsize = (15, 3))  # Create a figure and an axes.
ax.set_title('Fourier transform')

# YOUR CODE HERE
raise NotImplementedError()
```

In [ ]:
```
# This cell contains hidden tests.
```

In [ ]:
```
# This cell contains hidden tests.
```

Convolution with the Fourier transform

Convolution with the Fourier transform $g = \mathcal{F}^{-1}\{\mathcal{F}\{h\}\mathcal{F}\{f_{\text{true}}\}\}$

- Apply the transformed kernels in `list_h_fft` to `f_true`
- Return the absolute value of the inverse transform
- Save the outputs in the variable `list_g_dft` (as `list` of `numpy.array`)

Display the result

- Plot the outputs in `list_g_dft` in the same order of the parameter options in the axes `ax`
- Plot `f_true` in the axes `ax` (after `list_g_dft`)
- Show the legend in the axes `ax`

In [ ]:
```
fig, ax = plt.subplots(1, 1, figsize = (15, 3))  # Create a figure and an axes.
ax.set_title('Convolution with the Fourier transform')

# YOUR CODE HERE
raise NotImplementedError()
```

In [ ]:
```
# This cell contains hidden tests.
```

In [ ]:
```
# This cell contains tests.

for g_cov, g_dft in zip(list_g_cov, list_g_dft):
    np.testing.assert_almost_equal(g_cov[40:-40], g_dft[40:-40]) # with boundary
```

Question: Fourier transform

- What's the advantage to use the Fourier transform for convolution $\mathcal{F}\{g\} = \mathcal{F}\{h\}\mathcal{F}\{f_{\text{true}}\}$ for inverse problems?

YOUR ANSWER HERE