Image Analysis

Generated by Doxygen 1.9.1

1 Namespace Index 1
1.1 Packages 1

2 Hierarchical Index 3
2.1 Class Hierarchy 3

3 Class Index 5
3.1 Class List 5

4 File Index 7
4.1 File List 7

5 Namespace Documentation 9
5.1 imagetools Namespace Reference 9
5.2 imagetools.ml Namespace Reference 9
5.2.1 Function Documentation 9
5.2.1.1 gradient_descent() 9
5.3 imagetools.plotting Namespace Reference 10
5.3.1 Function Documentation 10
5.3.1.1 plot_image() 10
5.4 imagetools.signals Namespace Reference 11
5.4.1 Function Documentation 11
5.4.1.1 coeffs2img() 11
5.4.1.2 ComplexSoftThresh() 11
5.4.1.3 dwt2() 11
5.4.1.4 fft1c() 11
5.4.1.5 idwt2() 12
5.4.1.6 ifft1c() 12
5.4.1.7 img2coeffs() 12
5.4.1.8 iterative_soft_thresholding() 12
5.4.1.9 unstack_coeffs() 12
5.5 inverse_problems Namespace Reference 13
5.5.1 Variable Documentation 14
5.5.1.1 a_noisy_l2 14
5.5.1.2 a_outlier_l2 14
5.5.1.3 ax 14
5.5.1.4 aximg0 14
5.5.1.5 aximg1 14
5.5.1.6 B 14
5.5.1.7 B0 14
5.5.1.8 B_coefs 15
5.5.1.9 b_noisy_l2 15
5.5.1.10 b_outlier_l2 15
5.5.1.11 data 15

5.5.1.12 data_random_recon 15
5.5.1.13 data_unif_recon 15
5.5.1.14 eps 15
5.5.1.15 fig 15
5.5.1.16 figsize 16
5.5.1.17 file_list 16
5.5.1.18 grad 16
5.5.1.19 grad_l1 16
5.5.1.20 gt 16
5.5.1.21 label 16
5.5.1.22 lam 16
5.5.1.23 lr 17
5.5.1.24 mask_random 17
5.5.1.25 mask_unif 17
5.5.1.26 mse_random 17
5.5.1.27 mse_unif 17
5.5.1.28 n_iters 17
5.5.1.29 obj 17
5.5.1.30 obj_l1 17
5.5.1.31 obj_min 18
5.5.1.32 plotrange 18
5.5.1.33 river_img 18
5.5.1.34 river_img_dw 18
5.5.1.35 river_img_dw_th 18
5.5.1.36 river_img_recon 18
5.5.1.37 rng 18
5.5.1.38 sample_freq 19
5.5.1.39 sample_random 19
5.5.1.40 sample_unif 19
5.5.1.41 signal 19
5.5.1.42 signal_random 19
5.5.1.43 signal_random_recon 19
5.5.1.44 signal_unif 19
5.5.1.45 signal_unif_recon 19
5.5.1.46 start_index 20
5.5.1.47 th 20
5.5.1.48 title 20
5.5.1.49 x 20
5.5.1.50 x0 20
5.5.1.51 xlabel 20
5.5.1.52 y_noisy 20
5.5.1.53 y_outlier 20

5.5.1.54 ylabel 21
5.6 lgd Namespace Reference 21
5.6.1 Variable Documentation 22
5.6.1.1 adj_op_odl 22
5.6.1.2 args 22
5.6.1.3 ax 22
5.6.1.4 callback 22
5.6.1.5 checkpoint 23
5.6.1.6 checkpoint_filepath 23
5.6.1.7 choices 23
5.6.1.8 ckpt 23
5.6.1.9 data_fit 23
5.6.1.10 data_np 23
5.6.1.11 data_odl 23
5.6.1.12 device 24
5.6.1.13 f 24
5.6.1.14 fbp_np 24
5.6.1.15 fbp_odl 24
5.6.1.16 fbp_op_odl 24
5.6.1.17 fig 24
5.6.1.18 figsize 24
5.6.1.19 fwd_op_odl 25
5.6.1.20 g 25
5.6.1.21 geometry 25
5.6.1.22 grad 25
5.6.1.23 ground_truth 25
5.6.1.24 gt 25
5.6.1.25 img_size 25
5.6.1.26 L 26
5.6.1.27 lam 26
5.6.1.28 lgd_net 26
5.6.1.29 lgd_recon_np 26
5.6.1.30 loss 26
5.6.1.31 losses 26
5.6.1.32 mse_loss 26
5.6.1.33 niter 27
5.6.1.34 num_angles 27
5.6.1.35 num_epochs 27
5.6.1.36 num_learnable_params 27
5.6.1.37 op_norm 27
5.6.1.38 optimizer 27
5.6.1.39 parser 27

5.6.1.40 phantom_np . . . 28
5.6.1.41 phantom_odl . . . 28
5.6.1.42 reco_space . . . 28
5.6.1.43 recon . . . 28
5.6.1.44 reg_func . . . 28
5.6.1.45 required . . . 28
5.6.1.46 save_interval . . . 28
5.6.1.47 sigma . . . 29
5.6.1.48 start_idx . . . 29
5.6.1.49 step_size . . . 29
5.6.1.50 str . . . 29
5.6.1.51 tau . . . 29
5.6.1.52 type . . . 29
5.6.1.53 verbose_interval . . . 29
5.6.1.54 x_admm_np . . . 29
5.6.1.55 x_admm_odl . . . 30
5.6.1.56 x_init . . . 30
5.6.1.57 y . . . 30
5.7 segment Namespace Reference . . . 30
5.7.1 Variable Documentation . . . 31
5.7.1.1 ax . . . 31
5.7.1.2 cmap . . . 31
5.7.1.3 coins_img . . . 31
5.7.1.4 coins_labelled . . . 31
5.7.1.5 coins_marked . . . 31
5.7.1.6 coins_pre . . . 31
5.7.1.7 coins_segmented . . . 31
5.7.1.8 ct_img . . . 32
5.7.1.9 ct_labelled . . . 32
5.7.1.10 ct_mask . . . 32
5.7.1.11 ct_props . . . 32
5.7.1.12 ct_segmented . . . 32
5.7.1.13 ct_threshold . . . 32
5.7.1.14 desired_coins_idx . . . 32
5.7.1.15 fig . . . 33
5.7.1.16 figsize . . . 33
5.7.1.17 flowers_data . . . 33
5.7.1.18 flowers_flat . . . 33
5.7.1.19 flowers_img . . . 33
5.7.1.20 flowers_pre . . . 33
5.7.1.21 flowers_purple . . . 33
5.7.1.22 flowers_segmented . . . 33

5.7.1.23 km 34
5.7.1.24 lung_idx 34
5.7.1.25 se 34
5.7.1.26 verbose 34

6 Class Documentation 35

6.1 imagetools.ml.KMeans_Custom Class Reference 35
6.1.1 Detailed Description 35
6.1.2 Constructor & Destructor Documentation 35
6.1.2.1 __init__() 36
6.1.3 Member Function Documentation 36
6.1.3.1 assignment() 36
6.1.3.2 fit() 36
6.1.3.3 predict() 36
6.1.3.4 predict_cluster() 37
6.1.4 Member Data Documentation 37
6.1.4.1 centroids 37
6.1.4.2 K 37
6.2 lgd.LGD_net Class Reference 38
6.2.1 Constructor & Destructor Documentation 38
6.2.1.1 __init__() 39
6.2.2 Member Function Documentation 39
6.2.2.1 forward() 39
6.2.3 Member Data Documentation 39
6.2.3.1 adj_op_torch 39
6.2.3.2 fwd_op_torch 39
6.2.3.3 niter 39
6.2.3.4 prox 39
6.2.3.5 step_size 40
6.3 lgd.prox_net Class Reference 40
6.3.1 Constructor & Destructor Documentation 41
6.3.1.1 __init__() 41
6.3.2 Member Function Documentation 41
6.3.2.1 forward() 41
6.3.3 Member Data Documentation 41
6.3.3.1 act1 41
6.3.3.2 act2 42
6.3.3.3 conv1 42
6.3.3.4 conv2 42
6.3.3.5 conv3 42
6.3.3.6 pad 42

7 File Documentation 43

7.1 /home/jhughes2712/projects/image_analysis/jh2284/src/imagetools/__init__.py File Reference 43
7.2 /home/jhughes2712/projects/image_analysis/jh2284/src/imagetools/ml.py File Reference 43
7.2.1 Detailed Description 44
7.3 /home/jhughes2712/projects/image_analysis/jh2284/src/imagetools/plotting.py File Reference 44
7.3.1 Detailed Description 44
7.4 /home/jhughes2712/projects/image_analysis/jh2284/src/imagetools/signals.py File Reference 44
7.4.1 Detailed Description 45
7.5 /home/jhughes2712/projects/image_analysis/jh2284/src/inverse_problems.py File Reference 45
7.5.1 Detailed Description 46
7.6 /home/jhughes2712/projects/image_analysis/jh2284/src/lgd.py File Reference 47
7.6.1 Detailed Description 48
7.7 /home/jhughes2712/projects/image_analysis/jh2284/src/segment.py File Reference 48
7.7.1 Detailed Description 49

Index **51**

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

imagetools 9
imagetools.ml 9
imagetools.plotting 10
imagetools.signals 11
inverse_problems 13
lgd 21
segment 30

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

imagetools.ml.KMeans_Custom 35
nn.Module
    lgd.LGD_net 38
    lgd.prox_net 40

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

imagetools.ml.KMeans_Custom
Implements a class that handles KMeans Clustering 35
lgd.LGD_net 38
lgd.prox_net 40

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

/home/jhughes2712/projects/image_analysis/jh2284/src/inverse_problems.py
Script running code for questions 2.1, 2.2, 2.3, and 3.1 45
/home/jhughes2712/projects/image_analysis/jh2284/src/lgd.py
Script for training LGD 47
/home/jhughes2712/projects/image_analysis/jh2284/src/segment.py
Script running code for segmentation problems 48
/home/jhughes2712/projects/image_analysis/jh2284/src/imagetools/__init__.py 43
/home/jhughes2712/projects/image_analysis/jh2284/src/imagetools/ml.py
Module containing implementations of machine learning algorithms 43
/home/jhughes2712/projects/image_analysis/jh2284/src/imagetools/plotting.py
Module building on matplotlib to provide image-specific plotting functions 44
/home/jhughes2712/projects/image_analysis/jh2284/src/imagetools/signals.py
Module providing useful functions for image processing when employing Fourier and Wavelet transforms 44

Chapter 5

Namespace Documentation

5.1 imagetools Namespace Reference

Namespaces

- ml
- plotting
- signals

5.2 imagetools.ml Namespace Reference

Classes

- class KMeans_Custom

  Implements a class that handles KMeans Clustering.

Functions

- def gradient_descent (obj, grad, x0, obj_min, eps, lr, max_iters, filename)

  Implements vanilla gradient descent for scalar functions on R2, where the true global minimum is known.

5.2.1 Function Documentation

5.2.1.1 gradient_descent()

```
def imagetools.ml.gradient_descent (
              obj,
              grad,
              x0,
              obj_min,
              eps,
              lr,
              max_iters,
              filename )
```

Implements vanilla gradient descent for scalar functions on R2, where the true global minimum is known.

Parameters

obj	The objective function to minimise.
grad	The gradient of the objective function (R^2-->R^2)
x0	The initial iterate
obj_min	The minimal value of the objective function
eps	The threshold error for the stopping criterion
lr	The learning rate
max_iters	The maximum number of iterations to run before terminating
filename	Determines the name of the trajectory and loss plot file

Returns

x estimated argmin (in R^2) of the objective

5.3 imagetools.plotting Namespace Reference

Functions

- def plot_image (ax_sp, img, title, cmap, gt=None)
  Plots an image and compares to GT.

5.3.1 Function Documentation

5.3.1.1 plot_image()

```
def imagetools.plotting.plot_image (
              ax_sp,
              img,
              title,
              cmap,
              gt = None )
```

Plots an image and compares to GT.

```
@param ax_sp Specified matplotlib.pyplot.Axes object
@param img Image to be displayed
@param title Image title string
@param cmap String defining matplotlib colormap for image
@param gt Ground truth measurements to compare to
```

5.4 imagetools.signals Namespace Reference

Functions

- def iterative_soft_thresholding (data_sampled, lam, n_iters, gt=None)

  Implements ISTA in 1D.
- def fft1c (x)
- def ifft1c (y)
- def ComplexSoftThresh (y, lam)
- def coeffs2img (LL, coeffs)
- def img2coeffs (Wim, levels=4)
- def unstack_coeffs (Wim)
- def dwt2 (im)
- def idwt2 (Wim)

5.4.1 Function Documentation

5.4.1.1 coeffs2img()

```
def imagetools.signals.coeffs2img (
              LL,
              coeffs )
```

5.4.1.2 ComplexSoftThresh()

```
def imagetools.signals.ComplexSoftThresh (
              y,
              lam )
```

5.4.1.3 dwt2()

```
def imagetools.signals.dwt2 (
              im )
```

5.4.1.4 fft1c()

```
def imagetools.signals.fft1c (
              x )
```

5.4.1.5 idwt2()

```
def imagetools.signals.idwt2 (
              Wim )
```

5.4.1.6 ifft1c()

```
def imagetools.signals.ifft1c (
              y )
```

5.4.1.7 img2coeffs()

```
def imagetools.signals.img2coeffs (
              Wim,
              levels = 4 )
```

5.4.1.8 iterative_soft_thresholding()

```
def imagetools.signals.iterative_soft_thresholding (
              data_sampled,
              lam,
              n_iters,
              gt = None )
```

Implements ISTA in 1D.

```
@param data_sampled The subsampled measurement vector
@param lam The soft-thresholding (regularisation) parameter
@param n_iters Number of iterations
@param gt Ground truth measurements to compare to
@return data Approximated reconstructed measurements
@return mse_values Series of mse values per iteration to enable
performance plot
```

5.4.1.9 unstack_coeffs()

```
def imagetools.signals.unstack_coeffs (
              Wim )
```

5.5 inverse_problems Namespace Reference

Variables

- file_list = f.read().split("\n")[:-1]
- y_noisy = np.array([float(x) for x in file_list])
- y_outlier = np.array([float(x) for x in file_list])
- x = np.arange(len(y_noisy))
- a_noisy_l2 = np.sum(x ∗ y_noisy) / np.sum(x ∗ x)
- b_noisy_l2 = np.mean(y_noisy) - a_noisy_l2 ∗ np.mean(x)
- a_outlier_l2 = np.sum(x ∗ y_outlier) / np.sum(x ∗ x)
- b_outlier_l2 = np.mean(y_outlier) - a_outlier_l2 ∗ np.mean(x)
- fig
- ax
- figsize
- list B_coefs = [ ]
- obj_l1 = lambda B: np.sum(np.abs(B[0] ∗ x + B[1] - y))
- grad_l1
- B0 = np.array([0.1, 0.1])
- float lr = 0.0001
- B = gradient_descent(obj_l1, grad_l1, B0, 0, 0.01, lr, 100, "gd_l1")
- rng = np.random.default_rng(seed=42)
- signal = np.zeros(100)
- data = fft1c(signal)
- int sample_freq = 4
- mask_random = rng.uniform(0, 1, 100)
- mask_unif = np.zeros(100)
- start_index = rng.integers(0, sample_freq)
- sample_random = mask_random ∗ data
- sample_unif = mask_unif ∗ data
- int signal_random = ifft1c(sample_random) ∗ sample_freq
- int signal_unif = ifft1c(sample_unif) ∗ sample_freq
- data_random_recon
- mse_random
- lam
- n_iters = 1000
- gt
- data_unif_recon
- mse_unif
- int signal_random_recon = ifft1c(data_random_recon) ∗ sample_freq
- int signal_unif_recon = ifft1c(data_unif_recon) ∗ sample_freq
- label
- title
- xlabel
- ylabel
- river_img = skimage.io.imread("data/river_side.jpeg")
- river_img_dw = dwt2(river_img)
- river_img_recon = idwt2(river_img_dw)
- th = np.quantile(abs(river_img_dw), 0.85)
- river_img_dw_th = abs(river_img_dw) > th
- tuple plotrange
- aximg0 = ax[0].imshow(abs(river_img_dw), vmin=plotrange[0], vmax=plotrange[1])
- aximg1 = ax[1].imshow(abs(river_img_dw_th), cmap="grey")
- float obj = lambda x: 0.5 ∗ (x[0] ∗∗ 2) + (x[1] ∗∗ 2)
- grad = lambda x: np.array([x[0], 2.0 ∗ x[1]])
- x0 = np.array([1.0, 1.0])
- float eps = 0.01
- float obj_min = 0.0

5.5.1 Variable Documentation

5.5.1.1 a_noisy_l2

```
inverse_problems.a_noisy_l2 = np.sum(x * y_noisy) / np.sum(x * x)
```

5.5.1.2 a_outlier_l2

```
inverse_problems.a_outlier_l2 = np.sum(x * y_outlier) / np.sum(x * x)
```

5.5.1.3 ax

```
inverse_problems.ax
```

5.5.1.4 aximg0

```
inverse_problems.aximg0 = ax[0].imshow(abs(river_img_dw), vmin=plotrange[0], vmax=plotrange[1])
```

5.5.1.5 aximg1

```
inverse_problems.aximg1 = ax[1].imshow(abs(river_img_dw_th), cmap="grey")
```

5.5.1.6 B

```
inverse_problems.B = gradient_descent(obj_l1, grad_l1, B0, 0, 0.01, lr, 100, "gd_l1")
```

5.5.1.7 B0

```
inverse_problems.B0 = np.array([0.1, 0.1])
```

5.5.1.8 B_coefs

```
list inverse_problems.B_coefs = []
```

5.5.1.9 b_noisy_l2

```
inverse_problems.b_noisy_l2 = np.mean(y_noisy) - a_noisy_l2 * np.mean(x)
```

5.5.1.10 b_outlier_l2

```
inverse_problems.b_outlier_l2 = np.mean(y_outlier) - a_outlier_l2 * np.mean(x)
```

5.5.1.11 data

```
inverse_problems.data = fft1c(signal)
```

5.5.1.12 data_random_recon

```
inverse_problems.data_random_recon
```

5.5.1.13 data_unif_recon

```
inverse_problems.data_unif_recon
```

5.5.1.14 eps

```
float inverse_problems.eps = 0.01
```

5.5.1.15 fig

```
inverse_problems.fig
```

5.5.1.16 figsize

```
inverse_problems.figsize
```

5.5.1.17 file_list

```
inverse_problems.file_list = f.read().split("\n")[:-1]
```

5.5.1.18 grad

```
inverse_problems.grad = lambda x:  np.array([x[0], 2.0 * x[1]])
```

5.5.1.19 grad_l1

```
inverse_problems.grad_l1
```

Initial value:

```
=  lambda B: np.array(
        [
            np.dot(x, np.sign(B[0] * x + B[1] - y)),
            np.sum(np.sign(B[0] * x + B[1] - y)),
        ]
    )
```

5.5.1.20 gt

```
inverse_problems.gt
```

5.5.1.21 label

```
inverse_problems.label
```

5.5.1.22 lam

```
inverse_problems.lam
```

5.5.1.23 lr

```
float inverse_problems.lr = 0.0001
```

5.5.1.24 mask_random

```
inverse_problems.mask_random = rng.uniform(0, 1, 100)
```

5.5.1.25 mask_unif

```
inverse_problems.mask_unif = np.zeros(100)
```

5.5.1.26 mse_random

```
inverse_problems.mse_random
```

5.5.1.27 mse_unif

```
inverse_problems.mse_unif
```

5.5.1.28 n_iters

```
int inverse_problems.n_iters = 1000
```

5.5.1.29 obj

```
float inverse_problems.obj = lambda x:  0.5 * (x[0] ** 2) + (x[1] ** 2)
```

5.5.1.30 obj_l1

```
inverse_problems.obj_l1 = lambda B: np.sum(np.abs(B[0] * x + B[1] - y))
```

5.5.1.31 obj_min

```
float inverse_problems.obj_min = 0.0
```

5.5.1.32 plotrange

```
tuple inverse_problems.plotrange
```

Initial value:

```
= (
    np.quantile(abs(river_img_dw), 0.005),
    np.quantile(abs(river_img_dw), 0.995),
)
```

5.5.1.33 river_img

```
inverse_problems.river_img = skimage.io.imread("data/river_side.jpeg")
```

5.5.1.34 river_img_dw

```
inverse_problems.river_img_dw = dwt2(river_img)
```

5.5.1.35 river_img_dw_th

```
inverse_problems.river_img_dw_th = abs(river_img_dw) > th
```

5.5.1.36 river_img_recon

```
inverse_problems.river_img_recon = idwt2(river_img_dw)
```

5.5.1.37 rng

```
inverse_problems.rng = np.random.default_rng(seed=42)
```

5.5.1.38 sample_freq

```
int inverse_problems.sample_freq = 4
```

5.5.1.39 sample_random

```
inverse_problems.sample_random = mask_random * data
```

5.5.1.40 sample_unif

```
inverse_problems.sample_unif = mask_unif * data
```

5.5.1.41 signal

```
inverse_problems.signal = np.zeros(100)
```

5.5.1.42 signal_random

```
int inverse_problems.signal_random = ifft1c(sample_random) * sample_freq
```

5.5.1.43 signal_random_recon

```
int inverse_problems.signal_random_recon = ifft1c(data_random_recon) * sample_freq
```

5.5.1.44 signal_unif

```
int inverse_problems.signal_unif = ifft1c(sample_unif) * sample_freq
```

5.5.1.45 signal_unif_recon

```
int inverse_problems.signal_unif_recon = ifft1c(data_unif_recon) * sample_freq
```

5.5.1.46 start_index

```
inverse_problems.start_index = rng.integers(0, sample_freq)
```

5.5.1.47 th

```
inverse_problems.th = np.quantile(abs(river_img_dw), 0.85)
```

5.5.1.48 title

```
inverse_problems.title
```

5.5.1.49 x

```
inverse_problems.x = np.arange(len(y_noisy))
```

5.5.1.50 x0

```
inverse_problems.x0 = np.array([1.0, 1.0])
```

5.5.1.51 xlabel

```
inverse_problems.xlabel
```

5.5.1.52 y_noisy

```
inverse_problems.y_noisy = np.array([float(x) for x in file_list])
```

5.5.1.53 y_outlier

```
inverse_problems.y_outlier = np.array([float(x) for x in file_list])
```

5.5.1.54 ylabel

```
inverse_problems.ylabel
```

5.6 lgd Namespace Reference

Classes

- class prox_net
- class LGD_net

Variables

- parser = argparse.ArgumentParser()
- type
- str
- choices
- required
- args = parser.parse_args()
- int img_size = 256
- reco_space
- int num_angles = 30
- geometry = odl.tomo.parallel_beam_geometry(reco_space, num_angles=num_angles)
- fwd_op_odl = odl.tomo.RayTransform(reco_space, geometry)
- fbp_op_odl
- adj_op_odl = fwd_op_odl.adjoint
- phantom_odl = odl.phantom.shepp_logan(reco_space, modified=True)
- data_odl = fwd_op_odl(phantom_odl)
- fbp_odl = fbp_op_odl(data_odl)
- phantom_np = phantom_odl.__array__()
- fbp_np = fbp_odl.__array__()
- data_np = data_odl.__array__()
- fig
- ax
- figsize
- gt
- grad = odl.Gradient(reco_space)
- L = odl.BroadcastOperator(fwd_op_odl, grad)
- data_fit = odl.solvers.L2NormSquared(fwd_op_odl.range).translated(data_odl)
- float lam = 0.015
- float reg_func = lam ∗ odl.solvers.L1Norm(grad.range)
- g = odl.solvers.SeparableSum(data_fit, reg_func)
- f = odl.solvers.ZeroFunctional(L.domain)
- float op_norm = 1.1 ∗ odl.power_method_opnorm(L, maxiter=20)
- int niter = 200
- float sigma = 2.0
- float tau = sigma / op_norm∗∗2
- callback
- x_admm_odl = L.domain.zero()
- x_admm_np = x_admm_odl.__array__()
- tuple device

- int step_size = 1 / op_norm
- lgd_net = LGD_net().to(device)
- num_learnable_params
- tuple y
- tuple x_init
- tuple ground_truth
- mse_loss = torch.nn.MSELoss()
- optimizer = torch.optim.Adam(lgd_net.parameters(), lr=1e-4)
- int num_epochs = 2000
- ckpt = torch.load("outputs/weights_1990.pth", map_location=device)
- int start_idx = 1989
- losses = ckpt["losses"]
- int save_interval = 1
- int verbose_interval = 1
- recon = lgd_net(y, x_init)
- loss = mse_loss(recon, ground_truth)
- dictionary checkpoint
- string checkpoint_filepath = f"outputs/weights_{epoch+1:03d}.pth"
- tuple lgd_recon_np

5.6.1 Variable Documentation

5.6.1.1 adj_op_odl

```
lgd.adj_op_odl = fwd_op_odl.adjoint
```

5.6.1.2 args

```
lgd.args = parser.parse_args()
```

5.6.1.3 ax

```
lgd.ax
```

5.6.1.4 callback

```
lgd.callback
```

Initial value:

```
=  odl.solvers.CallbackPrintIteration(
    step=10
) & odl.solvers.CallbackShow(step=10)
```

5.6.1.5 checkpoint

```
dictionary lgd.checkpoint
```

Initial value:

```
=  {
            "epoch":  epoch + 1,
            "state_dict":  lgd_net.state_dict(),
            "optimizer":  optimizer.state_dict(),
            "losses":  losses,
        }
```

5.6.1.6 checkpoint_filepath

```
string lgd.checkpoint_filepath = f"outputs/weights_{epoch+1:03d}.pth"
```

5.6.1.7 choices

```
lgd.choices
```

5.6.1.8 ckpt

```
lgd.ckpt = torch.load("outputs/weights_1990.pth", map_location=device)
```

5.6.1.9 data_fit

```
lgd.data_fit = odl.solvers.L2NormSquared(fwd_op_odl.range).translated(data_odl)
```

5.6.1.10 data_np

```
lgd.data_np = data_odl.__array__()
```

5.6.1.11 data_odl

```
lgd.data_odl = fwd_op_odl(phantom_odl)
```

5.6.1.12 device

```
tuple lgd.device
```

Initial value:

```
= (
    torch.device("cuda") if torch.cuda.is_available() else torch.device("cpu")
)
```

5.6.1.13 f

```
lgd.f = odl.solvers.ZeroFunctional(L.domain)
```

5.6.1.14 fbp_np

```
lgd.fbp_np = fbp_odl.__array__()
```

5.6.1.15 fbp_odl

```
lgd.fbp_odl = fbp_op_odl(data_odl)
```

5.6.1.16 fbp_op_odl

```
lgd.fbp_op_odl
```

Initial value:

```
= odl.tomo.fbp_op(
    fwd_op_odl, filter_type="Ram-Lak", frequency_scaling=0.6
)
```

5.6.1.17 fig

```
lgd.fig
```

5.6.1.18 figsize

```
lgd.figsize
```

5.6.1.19 fwd_op_odl

```
lgd.fwd_op_odl = odl.tomo.RayTransform(reco_space, geometry)
```

5.6.1.20 g

```
lgd.g = odl.solvers.SeparableSum(data_fit, reg_func)
```

5.6.1.21 geometry

```
lgd.geometry = odl.tomo.parallel_beam_geometry(reco_space, num_angles=num_angles)
```

5.6.1.22 grad

```
lgd.grad = odl.Gradient(reco_space)
```

5.6.1.23 ground_truth

```
tuple lgd.ground_truth
```

Initial value:

```
= (
    torch.from_numpy(phantom_np).to(device).unsqueeze(0)
)
```

5.6.1.24 gt

```
lgd.gt
```

5.6.1.25 img_size

```
int lgd.img_size = 256
```

5.6.1.26 L

```
lgd.L = odl.BroadcastOperator(fwd_op_odl, grad)
```

5.6.1.27 lam

```
float lgd.lam = 0.015
```

5.6.1.28 lgd_net

```
lgd.lgd_net = LGD_net().to(device)
```

5.6.1.29 lgd_recon_np

```
tuple lgd.lgd_recon_np
```

Initial value:

```
=  (
    recon.detach().cpu().numpy().squeeze()
)
```

5.6.1.30 loss

```
lgd.loss = mse_loss(recon, ground_truth)
```

5.6.1.31 losses

```
list lgd.losses = ckpt["losses"]
```

5.6.1.32 mse_loss

```
lgd.mse_loss = torch.nn.MSELoss()
```

5.6.1.33 niter

```
lgd.niter = 200
```

5.6.1.34 num_angles

```
int lgd.num_angles = 30
```

5.6.1.35 num_epochs

```
int lgd.num_epochs = 2000
```

5.6.1.36 num_learnable_params

```
lgd.num_learnable_params
```

Initial value:

```
=  sum(
    p.numel() for p in lgd_net.parameters() if p.requires_grad
)
```

5.6.1.37 op_norm

```
float lgd.op_norm = 1.1 * odl.power_method_opnorm(L, maxiter=20)
```

5.6.1.38 optimizer

```
lgd.optimizer = torch.optim.Adam(lgd_net.parameters(), lr=1e-4)
```

5.6.1.39 parser

```
lgd.parser = argparse.ArgumentParser()
```

5.6.1.40 phantom_np

```
lgd.phantom_np = phantom_odl.__array__()
```

5.6.1.41 phantom_odl

```
lgd.phantom_odl = odl.phantom.shepp_logan(reco_space, modified=True)
```

5.6.1.42 reco_space

```
lgd.reco_space
```

Initial value:

```
=  odl.uniform_discr(
    min_pt=[-20, -20],
    max_pt=[20, 20],
    shape=[img_size, img_size],
    dtype="float32",
)
```

5.6.1.43 recon

```
lgd.recon = lgd_net(y, x_init)
```

5.6.1.44 reg_func

```
float lgd.reg_func = lam * odl.solvers.L1Norm(grad.range)
```

5.6.1.45 required

```
lgd.required
```

5.6.1.46 save_interval

```
int lgd.save_interval = 1
```

5.6.1.47 sigma

```
lgd.sigma = 2.0
```

5.6.1.48 start_idx

```
int lgd.start_idx = 1989
```

5.6.1.49 step_size

```
int lgd.step_size = 1 / op_norm
```

5.6.1.50 str

```
lgd.str
```

5.6.1.51 tau

```
lgd.tau = sigma / op_norm**2
```

5.6.1.52 type

```
lgd.type
```

5.6.1.53 verbose_interval

```
int lgd.verbose_interval = 1
```

5.6.1.54 x_admm_np

```
lgd.x_admm_np = x_admm_odl.__array__()
```

5.6.1.55 x_admm_odl

```
lgd.x_admm_odl = L.domain.zero()
```

5.6.1.56 x_init

```
tuple lgd.x_init
```

Initial value:

```
=  (
    torch.from_numpy(
        fbp_op_odl(y.detach().cpu().numpy().squeeze()).__array__()
    )
    .to(device)
    .unsqueeze(0)
)
```

5.6.1.57 y

```
tuple lgd.y
```

Initial value:

```
=  (
    torch.from_numpy(data_np).to(device).unsqueeze(0)
)
```

5.7 segment Namespace Reference

Variables

- fig
- ax
- figsize
- ct_img = skimage.io.imread("data/CT.png")
- ct_threshold = threshold_otsu(ct_img)
- ct_segmented = ct_img > ct_threshold
- se = disk(4)
- ct_mask = remove_small_objects(binary_opening(ct_segmented, se))
- ct_labelled = label(ct_mask == 0)
- ct_props = regionprops(ct_labelled)
- lung_idx = sorted(list(range(3)), key=lambda i: ct_props[i].area)[:2]
- cmap
- coins_img = skimage.io.imread("data/coins.png")
- coins_marked = coins_img.copy()
- coins_pre = rank.median(coins_img, np.ones((1, 7)))
- int coins_segmented = 1 - chan_vese(coins_pre, mu=0.1)
- coins_labelled = label(coins_segmented)
- list desired_coins_idx
- flowers_img = skimage.io.imread("data/noisy_flower.jpg")
- flowers_pre = denoise_tv_bregman(flowers_img, weight=0.5)
- km = KMeans_Custom(K=6)
- flowers_data = flowers_pre[::8, ::8, :].reshape(-1, 3)
- verbose
- flowers_flat = flowers_pre.reshape(-1, 3)
- flowers_segmented = km.predict_cluster(flowers_flat)
- int flowers_purple = 0

5.7.1 Variable Documentation

5.7.1.1 ax

```
segment.ax
```

5.7.1.2 cmap

```
segment.cmap
```

5.7.1.3 coins_img

```
tuple segment.coins_img = skimage.io.imread("data/coins.png")
```

5.7.1.4 coins_labelled

```
segment.coins_labelled = label(coins_segmented)
```

5.7.1.5 coins_marked

```
segment.coins_marked = coins_img.copy()
```

5.7.1.6 coins_pre

```
segment.coins_pre = rank.median(coins_img, np.ones((1, 7)))
```

5.7.1.7 coins_segmented

```
segment.coins_segmented = 1 - chan_vese(coins_pre, mu=0.1)
```

5.7.1.8 ct_img

```
segment.ct_img = skimage.io.imread("data/CT.png")
```

5.7.1.9 ct_labelled

```
float segment.ct_labelled = label(ct_mask == 0)
```

5.7.1.10 ct_mask

```
segment.ct_mask = remove_small_objects(binary_opening(ct_segmented, se))
```

5.7.1.11 ct_props

```
segment.ct_props = regionprops(ct_labelled)
```

5.7.1.12 ct_segmented

```
segment.ct_segmented = ct_img > ct_threshold
```

5.7.1.13 ct_threshold

```
segment.ct_threshold = threshold_otsu(ct_img)
```

5.7.1.14 desired_coins_idx

```
list segment.desired_coins_idx
```

Initial value:

```
=  [
    coins_labelled[50, 40],
    coins_labelled[125, 100],
    coins_labelled[200, 160],
    coins_labelled[275, 240],
]
```

5.7.1.15 fig

```
segment.fig
```

5.7.1.16 figsize

```
segment.figsize
```

5.7.1.17 flowers_data

```
segment.flowers_data = flowers_pre[::8, ::8, :].reshape(-1, 3)
```

5.7.1.18 flowers_flat

```
segment.flowers_flat = flowers_pre.reshape(-1, 3)
```

5.7.1.19 flowers_img

```
segment.flowers_img = skimage.io.imread("data/noisy_flower.jpg")
```

5.7.1.20 flowers_pre

```
segment.flowers_pre = denoise_tv_bregman(flowers_img, weight=0.5)
```

5.7.1.21 flowers_purple

```
segment.flowers_purple = 0
```

5.7.1.22 flowers_segmented

```
segment.flowers_segmented = km.predict_cluster(flowers_flat)
```

5.7.1.23 km

```
segment.km = KMeans_Custom(K=6)
```

5.7.1.24 lung_idx

```
segment.lung_idx = sorted(list(range(3)), key=lambda i:  ct_props[i].area)[:2]
```

5.7.1.25 se

```
segment.se = disk(4)
```

5.7.1.26 verbose

```
segment.verbose
```

Chapter 6

Class Documentation

6.1 imagetools.ml.KMeans_Custom Class Reference

Implements a class that handles KMeans Clustering.

Public Member Functions

- def __init__ (self, K)
- def assignment (self, data)

  Assigns a matrix where each row represents the assigned centroid.
- def fit (self, data, verbose=10, limit=1e5)

  Fits the KMeans clusters.
- def predict (self, data)

  Returns an array whose rows are the closest centroid to each datum.
- def predict_cluster (self, data)

  Returns an array whose entries are the closest centroid to each datum, mapped to a single identifying integer.

Public Attributes

- K
- centroids

6.1.1 Detailed Description

Implements a class that handles KMeans Clustering.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 __init__()

```
def imagetools.ml.KMeans_Custom.__init__ (
              self,
              K )
```

6.1.3 Member Function Documentation

6.1.3.1 assignment()

```
def imagetools.ml.KMeans_Custom.assignment (
              self,
              data )
```

Assigns a matrix where each row represents the assigned centroid.

Parameters

data	The data to assign labels to. Must be an (N, P) array.

Returns

labels

6.1.3.2 fit()

```
def imagetools.ml.KMeans_Custom.fit (
              self,
              data,
              verbose = 10,
              limit = 1e5 )
```

Fits the KMeans clusters.

```
@details Uses the KMeans++ algorithm to initialise centroids,
and then iteratively updates them using Lloyd’s algorithm.

@param data The data to assign labels to. Must be an (N, P) array.
@param verbose How often to print out number of reassigned labels.
@param limit Maximum number of iterations before stopping. Prevents
an indefinite loop.
```

6.1.3.3 predict()

```
def imagetools.ml.KMeans_Custom.predict (
              self,
              data )
```

Returns an array whose rows are the closest centroid to each datum.

Parameters

data	The data to assign labels to. Must be an (N, P) array.

Returns

predictions (N, P) array of centroids

6.1.3.4 predict_cluster()

```
def imagetools.ml.KMeans_Custom.predict_cluster (
              self,
              data )
```

Returns an array whose entries are the closest centroid to each datum, mapped to a single identifying integer.

Parameters

data	The data to assign labels to. Must be an (N, P) array.

Returns

predictions (N, 1) array of predicted clusters

6.1.4 Member Data Documentation

6.1.4.1 centroids

```
imagetools.ml.KMeans_Custom.centroids
```

6.1.4.2 K

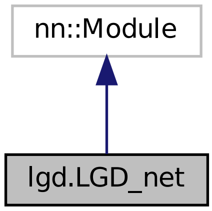
```
imagetools.ml.KMeans_Custom.K
```

The documentation for this class was generated from the following file:

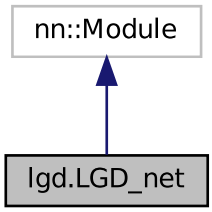
- /home/jhughes2712/projects/image_analysis/jh2284/src/imagetools/ml.py

6.2 lgd.LGD_net Class Reference

Inheritance diagram for lgd.LGD_net:



Collaboration diagram for lgd.LGD_net:



Public Member Functions

- def __init__ (self, niter=5, step_size=step_size)
- def forward (self, y, x_init)

Public Attributes

- niter
- prox
- step_size
- fwd_op_torch
- adj_op_torch

6.2.1 Constructor & Destructor Documentation

6.2.1.1 __init__()

```
def lgd.LGD_net.__init__ (
              self,
              niter = 5,
              step_size = step_size )
```

6.2.2 Member Function Documentation

6.2.2.1 forward()

```
def lgd.LGD_net.forward (
              self,
              y,
              x_init )
```

6.2.3 Member Data Documentation

6.2.3.1 adj_op_torch

```
lgd.LGD_net.adj_op_torch
```

6.2.3.2 fwd_op_torch

```
lgd.LGD_net.fwd_op_torch
```

6.2.3.3 niter

```
lgd.LGD_net.niter
```

6.2.3.4 prox

```
lgd.LGD_net.prox
```

6.2.3.5 step_size

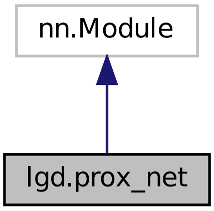
`lgd.LGD_net.step_size`

The documentation for this class was generated from the following file:

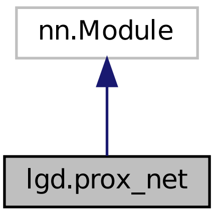
- /home/jhughes2712/projects/image_analysis/jh2284/src/lgd.py

6.3 lgd.prox_net Class Reference

Inheritance diagram for lgd.prox_net:



Collaboration diagram for lgd.prox_net:



Public Member Functions

- def __init__ (self, n_in_channels=2, n_out_channels=1, n_filters=32, kernel_size=3)
- def forward (self, x, u)

Public Attributes

- pad
- conv1
- conv2
- conv3
- act1
- act2

6.3.1 Constructor & Destructor Documentation

6.3.1.1 __init__()

```
def lgd.prox_net.__init__ (
            self,
            n_in_channels = 2,
            n_out_channels = 1,
            n_filters = 32,
            kernel_size = 3 )
```

6.3.2 Member Function Documentation

6.3.2.1 forward()

```
def lgd.prox_net.forward (
            self,
            x,
            u )
```

############ YOUR CODE HERE

6.3.3 Member Data Documentation

6.3.3.1 act1

```
lgd.prox_net.act1
```

6.3.3.2 act2

```
lgd.prox_net.act2
```

6.3.3.3 conv1

```
lgd.prox_net.conv1
```

6.3.3.4 conv2

```
lgd.prox_net.conv2
```

6.3.3.5 conv3

```
lgd.prox_net.conv3
```

6.3.3.6 pad

```
lgd.prox_net.pad
```

The documentation for this class was generated from the following file:

- /home/jhughes2712/projects/image_analysis/jh2284/src/lgd.py

Chapter 7

File Documentation

7.1 /home/jhughes2712/projects/image_↩ analysis/jh2284/src/imagetools/__init__.py File Reference

Namespaces

- imagetools

7.2 /home/jhughes2712/projects/image_↩ analysis/jh2284/src/imagetools/ml.py File Reference

Module containing implementations of machine learning algorithms.

Classes

- class imagetools.ml.KMeans_Custom

  Implements a class that handles KMeans Clustering.

Namespaces

- imagetools.ml

Functions

- def imagetools.ml.gradient_descent (obj, grad, x0, obj_min, eps, lr, max_iters, filename)

  Implements vanilla gradient descent for scalar functions on R2, where the true global minimum is known.

7.2.1 Detailed Description

Module containing implementations of machine learning algorithms.

Uses numpy to implement KMeans and Gradient Descent, in a way which is not as optimised as, say, scikit-learn, but is readable.

Author

Created by J. Hughes on 8th June 2024.

7.3 /home/jhughes2712/projects/image_↩ analysis/jh2284/src/imagetools/plotting.py File Reference

Module building on matplotlib to provide image-specific plotting functions.

Namespaces

- imagetools.plotting

Functions

- def imagetools.plotting.plot_image (ax_sp, img, title, cmap, gt=None)

  Plots an image and compares to GT.

7.3.1 Detailed Description

Module building on matplotlib to provide image-specific plotting functions.

Author

Created by J. Hughes on 8th June 2024.

7.4 /home/jhughes2712/projects/image_↩ analysis/jh2284/src/imagetools/signals.py File Reference

Module providing useful functions for image processing when employing Fourier and Wavelet transforms.

Namespaces

- imagetools.signals

Functions

- def imagetools.signals.iterative_soft_thresholding (data_sampled, lam, n_iters, gt=None)

  Implements ISTA in 1D.
- def imagetools.signals.fft1c (x)
- def imagetools.signals.ifft1c (y)
- def imagetools.signals.ComplexSoftThresh (y, lam)
- def imagetools.signals.coeffs2img (LL, coeffs)
- def imagetools.signals.img2coeffs (Wim, levels=4)
- def imagetools.signals.unstack_coeffs (Wim)
- def imagetools.signals.dwt2 (im)
- def imagetools.signals.idwt2 (Wim)

7.4.1 Detailed Description

Module providing useful functions for image processing when employing Fourier and Wavelet transforms.

Many of the functions are taken from the 'helper' module provided on the Image Analysis course GitLab page.

Author

Created by J. Hughes on 8th June 2024.

7.5 /home/jhughes2712/projects/image_analysis/jh2284/src/inverse_problems.py File Reference

Script running code for questions 2.1, 2.2, 2.3, and 3.1.

Namespaces

- inverse_problems

Variables

- inverse_problems.file_list = f.read().split("\n")[:-1]
- inverse_problems.y_noisy = np.array([float(x) for x in file_list])
- inverse_problems.y_outlier = np.array([float(x) for x in file_list])
- inverse_problems.x = np.arange(len(y_noisy))
- inverse_problems.a_noisy_l2 = np.sum(x * y_noisy) / np.sum(x * x)
- inverse_problems.b_noisy_l2 = np.mean(y_noisy) - a_noisy_l2 * np.mean(x)
- inverse_problems.a_outlier_l2 = np.sum(x * y_outlier) / np.sum(x * x)
- inverse_problems.b_outlier_l2 = np.mean(y_outlier) - a_outlier_l2 * np.mean(x)
- inverse_problems.fig
- inverse_problems.ax
- inverse_problems.figsize
- list inverse_problems.B_coefs = [ ]
- inverse_problems.obj_l1 = lambda B: np.sum(np.abs(B[0] * x + B[1] - y))
- inverse_problems.grad_l1
- inverse_problems.B0 = np.array([0.1, 0.1])

- float inverse_problems.lr = 0.0001
- inverse_problems.B = gradient_descent(obj_l1, grad_l1, B0, 0, 0.01, lr, 100, "gd_l1")
- inverse_problems.rng = np.random.default_rng(seed=42)
- inverse_problems.signal = np.zeros(100)
- inverse_problems.data = fft1c(signal)
- int inverse_problems.sample_freq = 4
- inverse_problems.mask_random = rng.uniform(0, 1, 100)
- inverse_problems.mask_unif = np.zeros(100)
- inverse_problems.start_index = rng.integers(0, sample_freq)
- inverse_problems.sample_random = mask_random * data
- inverse_problems.sample_unif = mask_unif * data
- int inverse_problems.signal_random = ifft1c(sample_random) * sample_freq
- int inverse_problems.signal_unif = ifft1c(sample_unif) * sample_freq
- inverse_problems.data_random_recon
- inverse_problems.mse_random
- inverse_problems.lam
- inverse_problems.n_iters = 1000
- inverse_problems.gt
- inverse_problems.data_unif_recon
- inverse_problems.mse_unif
- int inverse_problems.signal_random_recon = ifft1c(data_random_recon) * sample_freq
- int inverse_problems.signal_unif_recon = ifft1c(data_unif_recon) * sample_freq
- inverse_problems.label
- inverse_problems.title
- inverse_problems.xlabel
- inverse_problems.ylabel
- inverse_problems.river_img = skimage.io.imread("data/river_side.jpeg")
- inverse_problems.river_img_dw = dwt2(river_img)
- inverse_problems.river_img_recon = idwt2(river_img_dw)
- inverse_problems.th = np.quantile(abs(river_img_dw), 0.85)
- inverse_problems.river_img_dw_th = abs(river_img_dw) > th
- tuple inverse_problems.plotrange
- inverse_problems.aximg0 = ax[0].imshow(abs(river_img_dw), vmin=plotrange[0], vmax=plotrange[1])
- inverse_problems.aximg1 = ax[1].imshow(abs(river_img_dw_th), cmap="grey")
- float inverse_problems.obj = lambda x: 0.5 * (x[0] ** 2) + (x[1] ** 2)
- inverse_problems.grad = lambda x: np.array([x[0], 2.0 * x[1]])
- inverse_problems.x0 = np.array([1.0, 1.0])
- float inverse_problems.eps = 0.01
- float inverse_problems.obj_min = 0.0

7.5.1 Detailed Description

Script running code for questions 2.1, 2.2, 2.3, and 3.1.

This script solves various inverse problems related to signal and image processing, employing various iterative strategies and signal transforms.

Author

Created by J. Hughes on 8th June 2024.

7.6 /home/jhughes2712/projects/image_analysis/jh2284/src/lgd.py File Reference

Script for training LGD.

Classes

- class lgd.prox_net
- class lgd.LGD_net

Namespaces

- lgd

Variables

- lgd.parser = argparse.ArgumentParser()
- lgd.type
- lgd.str
- lgd.choices
- lgd.required
- lgd.args = parser.parse_args()
- int lgd.img_size = 256
- lgd.reco_space
- int lgd.num_angles = 30
- lgd.geometry = odl.tomo.parallel_beam_geometry(reco_space, num_angles=num_angles)
- lgd.fwd_op_odl = odl.tomo.RayTransform(reco_space, geometry)
- lgd.fbp_op_odl
- lgd.adj_op_odl = fwd_op_odl.adjoint
- lgd.phantom_odl = odl.phantom.shepp_logan(reco_space, modified=True)
- lgd.data_odl = fwd_op_odl(phantom_odl)
- lgd.fbp_odl = fbp_op_odl(data_odl)
- lgd.phantom_np = phantom_odl.__array__()
- lgd.fbp_np = fbp_odl.__array__()
- lgd.data_np = data_odl.__array__()
- lgd.fig
- lgd.ax
- lgd.figsize
- lgd.gt
- lgd.grad = odl.Gradient(reco_space)
- lgd.L = odl.BroadcastOperator(fwd_op_odl, grad)
- lgd.data_fit = odl.solvers.L2NormSquared(fwd_op_odl.range).translated(data_odl)
- float lgd.lam = 0.015
- float lgd.reg_func = lam ∗ odl.solvers.L1Norm(grad.range)
- lgd.g = odl.solvers.SeparableSum(data_fit, reg_func)
- lgd.f = odl.solvers.ZeroFunctional(L.domain)
- float lgd.op_norm = 1.1 ∗ odl.power_method_opnorm(L, maxiter=20)
- int lgd.niter = 200
- float lgd.sigma = 2.0
- float lgd.tau = sigma / op_norm∗∗2

- lgd.callback
- lgd.x_admm_odl = L.domain.zero()
- lgd.x_admm_np = x_admm_odl.__array__()
- tuple lgd.device
- int lgd.step_size = 1 / op_norm
- lgd.lgd_net = LGD_net().to(device)
- lgd.num_learnable_params
- tuple lgd.y
- tuple lgd.x_init
- tuple lgd.ground_truth
- lgd.mse_loss = torch.nn.MSELoss()
- lgd.optimizer = torch.optim.Adam(lgd_net.parameters(), lr=1e-4)
- int lgd.num_epochs = 2000
- lgd.ckpt = torch.load("outputs/weights_1990.pth", map_location=device)
- int lgd.start_idx = 1989
- lgd.losses = ckpt["losses"]
- int lgd.save_interval = 1
- int lgd.verbose_interval = 1
- lgd.recon = lgd_net(y, x_init)
- lgd.loss = mse_loss(recon, ground_truth)
- dictionary lgd.checkpoint
- string lgd.checkpoint_filepath = f"outputs/weights_{epoch+1:03d}.pth"
- tuple lgd.lgd_recon_np

7.6.1 Detailed Description

Script for training LGD.

Reconstructs CT images using FBP and ADMM, and compares to a data-driven LGD algorithm. Can be run in 'demo' mode, just performing the last 10 epochs of training from a checkpoint, or 'full' mode which runs all 2000 training epochs.

Author

Created by J. Hughes on 8th June 2024.

7.7 /home/jhughes2712/projects/image_analysis/jh2284/src/segment.py File Reference

Script running code for segmentation problems.

Namespaces

- segment

Variables

- segment.fig
- segment.ax
- segment.figsize
- segment.ct_img = skimage.io.imread("data/CT.png")
- segment.ct_threshold = threshold_otsu(ct_img)
- segment.ct_segmented = ct_img > ct_threshold
- segment.se = disk(4)
- segment.ct_mask = remove_small_objects(binary_opening(ct_segmented, se))
- segment.ct_labelled = label(ct_mask == 0)
- segment.ct_props = regionprops(ct_labelled)
- segment.lung_idx = sorted(list(range(3)), key=lambda i: ct_props[i].area)[:2]
- segment.cmap
- segment.coins_img = skimage.io.imread("data/coins.png")
- segment.coins_marked = coins_img.copy()
- segment.coins_pre = rank.median(coins_img, np.ones((1, 7)))
- int segment.coins_segmented = 1 - chan_vese(coins_pre, mu=0.1)
- segment.coins_labelled = label(coins_segmented)
- list segment.desired_coins_idx
- segment.flowers_img = skimage.io.imread("data/noisy_flower.jpg")
- segment.flowers_pre = denoise_tv_bregman(flowers_img, weight=0.5)
- segment.km = KMeans_Custom(K=6)
- segment.flowers_data = flowers_pre[::8, ::8, :].reshape(-1, 3)
- segment.verbose
- segment.flowers_flat = flowers_pre.reshape(-1, 3)
- segment.flowers_segmented = km.predict_cluster(flowers_flat)
- int segment.flowers_purple = 0

7.7.1 Detailed Description

Script running code for segmentation problems.

This script segments the three images given on the question sheet using three different segmentation algorithms.

Author

Created by J. Hughes on 8th June 2024.

Index

/home/jhughes2712/projects/image_analysis/jh2284/src/imagetools/__init__.py, 43
/home/jhughes2712/projects/image_analysis/jh2284/src/imagetools/ml.py, 43
/home/jhughes2712/projects/image_analysis/jh2284/src/imagetools/plotting.py, 44
/home/jhughes2712/projects/image_analysis/jh2284/src/imagetools/signals.py, 44
/home/jhughes2712/projects/image_analysis/jh2284/src/inverse_problems.py, 45
/home/jhughes2712/projects/image_analysis/jh2284/src/lgd.py, 47
/home/jhughes2712/projects/image_analysis/jh2284/src/segment.py, 48

__init__
- imagetools.ml.KMeans_Custom, 35
- lgd.LGD_net, 38
- lgd.prox_net, 41

a_noisy_l2
- inverse_problems, 14

a_outlier_l2
- inverse_problems, 14

act1
- lgd.prox_net, 41

act2
- lgd.prox_net, 41

adj_op_odl
- lgd, 22

adj_op_torch
- lgd.LGD_net, 39

args
- lgd, 22

assignment
- imagetools.ml.KMeans_Custom, 36

ax
- inverse_problems, 14
- lgd, 22
- segment, 31

aximg0
- inverse_problems, 14

aximg1
- inverse_problems, 14

B
- inverse_problems, 14

B0
- inverse_problems, 14

B_coefs
- inverse_problems, 14

b_noisy_l2
- inverse_problems, 15

b_outlier_l2
- inverse_problems, 15

callback
- lgd, 22

centroids
- imagetools.ml.KMeans_Custom, 37

checkpoint
- lgd, 22

checkpoint_filepath
- lgd, 23

choices
- lgd, 23

ckpt
- lgd, 23

cmap
- segment, 31

coeffs2img
- imagetools.signals, 11

coins_img
- segment, 31

coins_labelled
- segment, 31

coins_marked
- segment, 31

coins_pre
- segment, 31

coins_segmented
- segment, 31

ComplexSoftThresh
- imagetools.signals, 11

conv1
- lgd.prox_net, 42

conv2
- lgd.prox_net, 42

conv3
- lgd.prox_net, 42

ct_img
- segment, 31

ct_labelled
- segment, 32

ct_mask
- segment, 32

ct_props
- segment, 32

ct_segmented
- segment, 32

ct_threshold

segment, 32

data
inverse_problems, 15
data_fit
lgd, 23
data_np
lgd, 23
data_odl
lgd, 23
data_random_recon
inverse_problems, 15
data_unif_recon
inverse_problems, 15
desired_coins_idx
segment, 32
device
lgd, 23
dwt2
imagetools.signals, 11

eps
inverse_problems, 15

f
lgd, 24
fbp_np
lgd, 24
fbp_odl
lgd, 24
fbp_op_odl
lgd, 24
fft1c
imagetools.signals, 11
fig
inverse_problems, 15
lgd, 24
segment, 32
figsize
inverse_problems, 15
lgd, 24
segment, 33
file_list
inverse_problems, 16
fit
imagetools.ml.KMeans_Custom, 36
flowers_data
segment, 33
flowers_flat
segment, 33
flowers_img
segment, 33
flowers_pre
segment, 33
flowers_purple
segment, 33
flowers_segmented
segment, 33
forward
lgd.LGD_net, 39
lgd.prox_net, 41
fwd_op_odl
lgd, 24
fwd_op_torch
lgd.LGD_net, 39

g
lgd, 25
geometry
lgd, 25
grad
inverse_problems, 16
lgd, 25
grad_l1
inverse_problems, 16
gradient_descent
imagetools.ml, 9
ground_truth
lgd, 25
gt
inverse_problems, 16
lgd, 25

idwt2
imagetools.signals, 11
ifft1c
imagetools.signals, 12
imagetools, 9
imagetools.ml, 9
gradient_descent, 9
imagetools.ml.KMeans_Custom, 35
__init__, 35
assignment, 36
centroids, 37
fit, 36
K, 37
predict, 36
predict_cluster, 37
imagetools.plotting, 10
plot_image, 10
imagetools.signals, 11
coeffs2img, 11
ComplexSoftThresh, 11
dwt2, 11
fft1c, 11
idwt2, 11
ifft1c, 12
img2coeffs, 12
iterative_soft_thresholding, 12
unstack_coeffs, 12
img2coeffs
imagetools.signals, 12
img_size
lgd, 25
inverse_problems, 13
a_noisy_l2, 14
a_outlier_l2, 14
ax, 14

aximg0, 14
aximg1, 14
B, 14
B0, 14
B_coefs, 14
b_noisy_l2, 15
b_outlier_l2, 15
data, 15
data_random_recon, 15
data_unif_recon, 15
eps, 15
fig, 15
figsize, 15
file_list, 16
grad, 16
grad_l1, 16
gt, 16
label, 16
lam, 16
lr, 16
mask_random, 17
mask_unif, 17
mse_random, 17
mse_unif, 17
n_iters, 17
obj, 17
obj_l1, 17
obj_min, 17
plotrange, 18
river_img, 18
river_img_dw, 18
river_img_dw_th, 18
river_img_recon, 18
rng, 18
sample_freq, 18
sample_random, 19
sample_unif, 19
signal, 19
signal_random, 19
signal_random_recon, 19
signal_unif, 19
signal_unif_recon, 19
start_index, 19
th, 20
title, 20
x, 20
x0, 20
xlabel, 20
y_noisy, 20
y_outlier, 20
ylabel, 20

iterative_soft_thresholding
imagetools.signals, 12

K
imagetools.ml.KMeans_Custom, 37

km
segment, 33

L
lgd, 25

label
inverse_problems, 16

lam
inverse_problems, 16
lgd, 26

lgd, 21
adj_op_odl, 22
args, 22
ax, 22
callback, 22
checkpoint, 22
checkpoint_filepath, 23
choices, 23
ckpt, 23
data_fit, 23
data_np, 23
data_odl, 23
device, 23
f, 24
fbp_np, 24
fbp_odl, 24
fbp_op_odl, 24
fig, 24
figsize, 24
fwd_op_odl, 24
g, 25
geometry, 25
grad, 25
ground_truth, 25
gt, 25
img_size, 25
L, 25
lam, 26
lgd_net, 26
lgd_recon_np, 26
loss, 26
losses, 26
mse_loss, 26
niter, 26
num_angles, 27
num_epochs, 27
num_learnable_params, 27
op_norm, 27
optimizer, 27
parser, 27
phantom_np, 27
phantom_odl, 28
reco_space, 28
recon, 28
reg_func, 28
required, 28
save_interval, 28
sigma, 28
start_idx, 29
step_size, 29
str, 29

tau, 29
type, 29
verbose_interval, 29
x_admm_np, 29
x_admm_odl, 29
x_init, 30
y, 30
lgd.LGD_net, 38
__init__, 38
adj_op_torch, 39
forward, 39
fwd_op_torch, 39
niter, 39
prox, 39
step_size, 39
lgd.prox_net, 40
__init__, 41
act1, 41
act2, 41
conv1, 42
conv2, 42
conv3, 42
forward, 41
pad, 42
lgd_net
lgd, 26
lgd_recon_np
lgd, 26
loss
lgd, 26
losses
lgd, 26
lr
inverse_problems, 16
lung_idx
segment, 34

mask_random
inverse_problems, 17
mask_unif
inverse_problems, 17
mse_loss
lgd, 26
mse_random
inverse_problems, 17
mse_unif
inverse_problems, 17

n_iters
inverse_problems, 17
niter
lgd, 26
lgd.LGD_net, 39
num_angles
lgd, 27
num_epochs
lgd, 27
num_learnable_params
lgd, 27

obj
inverse_problems, 17
obj_l1
inverse_problems, 17
obj_min
inverse_problems, 17
op_norm
lgd, 27
optimizer
lgd, 27

pad
lgd.prox_net, 42
parser
lgd, 27
phantom_np
lgd, 27
phantom_odl
lgd, 28
plot_image
imagetools.plotting, 10
plotrange
inverse_problems, 18
predict
imagetools.ml.KMeans_Custom, 36
predict_cluster
imagetools.ml.KMeans_Custom, 37
prox
lgd.LGD_net, 39

reco_space
lgd, 28
recon
lgd, 28
reg_func
lgd, 28
required
lgd, 28
river_img
inverse_problems, 18
river_img_dw
inverse_problems, 18
river_img_dw_th
inverse_problems, 18
river_img_recon
inverse_problems, 18
rng
inverse_problems, 18

sample_freq
inverse_problems, 18
sample_random
inverse_problems, 19
sample_unif
inverse_problems, 19
save_interval
lgd, 28
se
segment, 34

segment, 30
  ax, 31
  cmap, 31
  coins_img, 31
  coins_labelled, 31
  coins_marked, 31
  coins_pre, 31
  coins_segmented, 31
  ct_img, 31
  ct_labelled, 32
  ct_mask, 32
  ct_props, 32
  ct_segmented, 32
  ct_threshold, 32
  desired_coins_idx, 32
  fig, 32
  figsize, 33
  flowers_data, 33
  flowers_flat, 33
  flowers_img, 33
  flowers_pre, 33
  flowers_purple, 33
  flowers_segmented, 33
  km, 33
  lung_idx, 34
  se, 34
  verbose, 34
sigma
  lgd, 28
signal
  inverse_problems, 19
signal_random
  inverse_problems, 19
signal_random_recon
  inverse_problems, 19
signal_unif
  inverse_problems, 19
signal_unif_recon
  inverse_problems, 19
start_idx
  lgd, 29
start_index
  inverse_problems, 19
step_size
  lgd, 29
  lgd.LGD_net, 39
str
  lgd, 29

tau
  lgd, 29
th
  inverse_problems, 20
title
  inverse_problems, 20
type
  lgd, 29

unstack_coeffs
  imagetools.signals, 12

verbose
  segment, 34
verbose_interval
  lgd, 29

x
  inverse_problems, 20
x0
  inverse_problems, 20
x_admm_np
  lgd, 29
x_admm_odl
  lgd, 29
x_init
  lgd, 30
xlabel
  inverse_problems, 20

y
  lgd, 30
y_noisy
  inverse_problems, 20
y_outlier
  inverse_problems, 20
ylabel
  inverse_problems, 20