pyM2aia

Jonas Cordes

Mar 15, 2024

CONTENTS

m2aia.Dataset module

class m2aia.Dataset.**BaseDataSet**(*images: List[ImzMLReader]*, *buffer_type: str*)

Bases: `object`

getitems(*indexes: List[int]*)

class m2aia.Dataset.**IonImageDataset**(*images: List[ImzMLReader]*, *centroids: List[float]*, *tolerance: float*, *tolerance_type: str = 'ppm'*, *buffer_type='memory'*, *transforms=None*)

Bases: *BaseDataSet*

get_tolerance(*c*)

getitems(*indexes: List[int]*)

make_buffered_image(*index*)

class m2aia.Dataset.**SpectrumDataset**(*images: ~typing.List[~m2aia.ImageIO.ImzMLReader]*, *labeled_images: ~typing.List[~numpy.array] | None = None*, *sampling_masks: ~typing.List[~numpy.array] | None = None*, *spectrum_mask_indices: ~typing.List[int] | None = None*, *tolerance: ~numpy.float32 | None = None*, *is_tolerance_in_ppm: bool = True*, *label_map: ~typing.Dict | None = None*, *shape: ~typing.Tuple | None = None*, *transform_data=None*, *transform_labels=None*, *buffer_type: str = 'memory'*, *reduce_function=<function mean>*, *shuffle=False*, *quiet_init=True*)

Bases: *BaseDataSet*

Dataset for accession individual spectra and class labels (optional) of multiple images (m2aia.ImzMLReader objects).

The aim of the SpectrumDataset is to provide convenient access to spectra of single or multiple ImzMLReaders. Two access strategy exist: 1) Spectral approach: a single spectrum is returned. 2) Spatio-spectral: a central spectrum and corresponding neighbors are returned.

To use multiple images a spectra depth of equal size for each image is required.

A label mask can be provided and is used to return labels for each accessed element.

To use the spatio-spectral approach, a shape element is required. The Dataset will then return the spectrum embedded in neighboring spectra, i.e. if the shape tuple is shape=(5,5) the shape of a data entry is [B,C,H,W], with batchsize as B = 1, spectrum depth as C = len(spectrum), width as W=5 and height as H=5 of the patch.

If no shape element was provided, the Dataset will return a single spectrum with shape [B=1,C].

If multiple elements of the Dataset should be queried at one, the SpectrumDataset.getitems(list_of_indices) returns a batch like object containing the elements. i.e. without a shape definition returned elements will have the shape [B=len(list_of_indices), C] and with shape=(5,5) the shape [B=len(list_of_indices),C,H=5,W=5]. This is used in m2aia.BatchGenerator.

Complete processing examples with focus on deep learning can be found on https://github.com/m2aia/pym2aia-examples

Example usage:

```
import m2aia as m2

I = m2.ImzMLReader("path/to/imzMl/file.imzML")
I.SetNormalization(m2.m2NormalizationTIC)
```

(continues on next page)

(continued from previous page)

```
I.SetIntensityTransformation(m2.m2IntensityTransformationSquareRoot)
I.Execute()

dataset = m2.SpectrumDataset([I], shuffle=True)
for X,Y in dataset():
    print("Spectrum", X.shape, "Class Labels", Y.shape)
    do_something(X,Y)
```

find_nearest_indices(*centroids: array*, *xaxis: array*)

find_subrange_indices(*xs*, *center_index*, *tolerance*, *is_ppm*)

getitems(*dataset_query_indices: List[int]*)

CHAPTER

ONE

M2AIA.GENERATORS MODULE

class m2aia.Generators.**BatchGenerator**(*dataset:* BaseDataSet, *batch_size: int*, *shuffle: bool = True*)

Bases: object

on_epoch_end()

CHAPTER

TWO

M2AIA.IMAGEIO MODULE

class m2aia.ImageIO.**ImzMLReader**(*imzML_path*, *baseline_correction: Literal['TopHat', 'Median', 'None'] = 'None'*, *baseline_correction_half_window_size: int = 50*, *normalization: Literal['TIC', 'Sum', 'Mean', 'Max', 'RMS', 'Internal', 'External', 'None'] = 'None'*, *smoothing: Literal['SavitzkyGolay', 'Gaussian', 'None'] = 'None'*, *smoothing_half_window_size: int = 2*, *intensity_transformation: Literal['Log2', 'Log10', 'SquareRoot', 'None'] = 'None'*, *pooling: Literal['Mean', 'Median', 'Maximum', 'Sum'] = 'Maximum'*)

Bases: `object`

Wrapper class for M2aia's imzML reader https://github.com/m2aia/m2aia.

Complete processing examples with focus on deep learning can be found on https://github.com/m2aia/pym2aia-examples

Example usage:

```
import m2aia as m2

I = m2.ImzMLReader("path/to/imzMl/file.imzML")
I.SetNormalization(m2.m2NormalizationTIC)
I.SetIntensityTransformation(m2.m2IntensityTransformationSquareRoot)
I.Execute()
ys_2 = I.GetMeanSpectrum()
i_2 = I.GetArray(imz, 75)
```

CheckHandle()

Check if the handle was initialized properly. To prevent this check from throwing an exception you must call Execute() once.

Raises

ReferenceError: is invalid file name and or not yet called Execute().

GetArray(*center*, *tol*, *dtype=<class 'numpy.float32'>*, *squeeze: bool = False*) → ndarray

Get the (ion) image data as numpy array. The pixel values are the pooled intensities (ie. pooling strategies like the 'Mean', 'Median', 'Maximum', or 'Sum') in the interval [center-tol, center+tol] of the spectra.

Returns

Numpy array of size [x,y,z] with dtype=dtype.

Parameters

- **center** – value on the x axis.
- **tol** – tolerance for query points on the x axis around the center.

- **dtype** – array element type [np.float32, np.float64].
- **squeeze** – Remove all dimensions if any is smaller or equals 1.

Raises
TypeError: Image pixel type is not one of [np.float32, np.float64]

GetImage(*center*, *tol*, *dtype=<class 'numpy.float32'>*) → Image

Get the (ion) image data as parameterized SimpleITK.Image.

`m2aia.ImzMLReader.GetArray()`

Returns
sitk.Image of size [x,y,z] with dtype=dtype.

Parameters

- **center** – value on the x axis.
- **tol** – tolerance for query points on the x axis around the center.
- **dtype** – array element type [np.float32, np.float64].
- **squeeze** – Remove all dimensions if any is smaller or equals 1.

Raises
TypeError: Image pixel type is not one of [np.float32, np.float64]

GetIndexArray() → ndarray

Get the index image data as numpy array. The pixel values are the spectrum IDs starting with 0. Use the mask array to identify valid spectrum IDs.

Access valid spectrum IDs:

```
indexImage = I.GetIndexArray()
maskImage = I.GetMaskArray()
validIndices = indexImage[maskImage > 0]
```

Returns
Numpy array of size [x,y,z] with dtype=np.uint32.

GetIndexImage()

Get the index image data as parameterized SimpleITK.Image.
This method calls `GetIndexArray()`

Returns
sitk.Image of size [x,y,z] with dtype=np.uint32.

GetIntensities(*index*, *ys=None*) → array

Query the y-axis (ys) values for a given spectrum id.

Parameters

- **index** – Id of a spectrum in the image.
- **ys** – By passing a np.array (dtype=np.float32) from external, this np.array can be reused and do not require an extra memory allocation. Otherwise a new array is created.

Returns
A list of two np.array elements [xs,ys]. xs = x-values; ys = y-values.

Raises
IndexError: if index is not in the range of valid spectra indices [0,self.number_of_spectra-1].
TypeError: if the ImzML file format is not continuous profile/centroid!

GetMaskArray() → ndarray

Get the mask image data as numpy array. The binary mask indicates valid spectra (pixel value >= 1) and background (pixel value == 0).

Returns
Numpy array of size [x,y,z] with dtype=np.ushort.

GetMaskImage() → Image

Get the mask image data as parameterized SimpleITK.Image. The pixel values indicate valid spectra (pixel value >= 1) and background (pixel value == 0).

Returns
sitk.Image of size [x,y,z] with dtype=np.ushort.

GetMaxSpectrum() → array

Get the overview spectrum (max over all spectra).

Returns
np.array with maximum intensity values of all spectra.

GetMeanSpectrum() → array

Get the overview spectrum (mean over all spectra).

Returns
np.array with mean intensity values of all spectra.

GetMetaData() → Dict[str, str]

Returns a dictionary of all meta data information retrieved by m2aia.

Returns
List of strings of meta data.

GetNormalizationArray(*type*) → ndarray

Get a normalization image data as numpy array.

Returns
Numpy array of size [x,y,z] with dtype=np.float64.

GetNormalizationImage(*type*) → Image

Get a normalization image data as parameterized SimpleITK.Image.

Returns
sitk.Image of size [x,y,z] with dtype=np.float64.

GetNumberOfSpectra() → int

Get the number of valid spectra in the image. This can be used to iterate over all spectra in the image using a for loop:

```
for i in range(GetNumberOfSpectra()):
    xs, ys = reader.GetSpectrum(i)
```

Returns
Number of valid spectra.

`GetOrigin()` → array

Get the image origin.

Returns
The origin in world coordinates of the image as numpy array of size [3] and dtype=np.float64.

`GetParametersAsFormattedString()`

Transform signal processing parameters into a fomatted string representation.

`GetShape()` → array

Get the shape of the image.

Returns
numpy array of size [3] for the x,y,z image dimensions (in number of pixels).

`GetSizeInBytesOfYAxisType()` → int

Get number of bytes used to store the intensity values.

Returns
The number of bytes.

`GetSpacing()` → array

Get the pixel spacing of the image

Returns
numpy array of size [3] and dtype=np.float64 for the pixel size in x,y,z dimension (in millimeter).

`GetSpectra`(*indices: List[int]*) → ndarray

Query a set of intensities by a list of indices. Only continuous imzML files.

Parameters
indices – List of Ids of spectra in the image.

Returns
a np.ndarray of shape [len(indices), self.depth].

Raises
IndexError: if index is not in the range of valid spectra indices [0,self.number_of_spectra-1].
TypeError: if the ImzML file format is not continuous profile/centroid!

`GetSpectrum`(*index*) → List[array]

Query the x-axis (xs) and y-axis (ys) values for a given spectrum id.

Parameters
index – Id of a spectrum in the image.

Returns
A list of two np.array elements [xs,ys]. xs = x-values; ys = y-values.

Raises
IndexError: if index is not in the range of valid spectra indices [0,self.number_of_spectra-1].

`GetSpectrumDepth`(*id*) → int

Get the size of the x axis for a specific spectrum of the image. This method is helpful for processed (centroid) imzML files. For continuous (centroid/profile) imzML files use GetXAxisDepth(self).

Parameters
id – Id of a spectrum in the image.

Returns
Number of x values.

GetSpectrumPosition(*id*) → array

Get the image index position for a given spectrum id.

Parameters
id – the id of a spectrum (may be different to the ids given in the imzML).

Returns
Position in index coordinates as numpy array of size [3] and dtype=np.int32.

GetSpectrumType() → str

Get the imzML type, i.e. continuous/processed profile/centroid.

Returns
The type of the imzML as string.

GetTolerance() → float32

Get the current tolerance value.

Returns
np.float32 The current tolerance value.

GetXAxis() → array

Get the x axis values (i.e. m/z values on the x axis).

Returns
np.array

GetXAxisDepth() → int

Get the size of the x axis. For processed imzML files, this value is the number of bins used to represent the x-axis.

Returns
Number of x values.

GetYDataType()

Return the intensity data type defined in the imzML file.

Returns
np.float32 or np.float64; if not defined return None

Load()

SetBaselineCorrection(*strategy: Literal['TopHat', 'Median', 'None']*, *half_window_size=50*)

Set the baseline correction strategy.

Parameters

- **strategy** – Set the basline correction strategy using one of the m2BaselineCorrection literals.
- **half_window_size** – 2*half_window_size + 1 spectrum points are used for BaselineCorrection.

SetIntensityTransformation(*strategy: Literal['Log2', 'Log10', 'SquareRoot', 'None']*)

Set the intensity transformation strategy.

Parameters
strategy – m2IntensityTransformation Set the intensity transformation strategy using one of the m2IntensityTransformation literals.

SetNormalization(*strategy: Literal['TIC', 'Sum', 'Mean', 'Max', 'RMS', 'Internal', 'External', 'None']*)

Set the normalization strategy.

Parameters

strategy – m2Normalization Set the normalization strategy using one of the m2Normalization literals.

SetPooling(*strategy: Literal['Mean', 'Median', 'Maximum', 'Sum']*)

Set the pooling strategy.

Parameters

strategy – m2Pooling Set the pooling strategy using one of the m2Pooling literals.

SetSmoothing(*strategy: Literal['SavitzkyGolay', 'Gaussian', 'None']*, *half_window_size=2*)

Set the spectrum smoothing strategy.

Parameters

- **strategy** – Set the smoothing strategy using one of the m2Smoothing literals.
- **half_window_size** – 2*half_window_size + 1 spectrum points used for smoothing.

SetTolerance(*tol: float32*)

Set the tolerance value.

Parameters

tol – np.float32 The tolerance value to be set.

SpectrumIterator()

Create a spectrum iterator/generator, yielding all valid spectra. This can be used to iterate over all spectra in the image using a for loop:

```
for i,xs,ys in reader.SpectrumIterator():
    ...
```

Returns

a triplet with (i=spectrum-id, xs=x-values, ys=y-values)

SpectrumRandomBatchIterator(*batch_size*)

Create a spectrum batch iterator/generator, yielding a batch of ys-values of spectra in a random order with repetitions.

Example:

```
for ys_batch in reader.SpectrumRandomBatchIterator(batch_size = N):
    ...
```

Returns

a np.array as batch of intensities with shape [batch_size, self.depth]

WriteContinuousCentroidImzML(*path: str*, *centroids*)

Given a list of centroids, write a continuous centroid imzML to the given path. Use 'SetTolerance' to define the range query for each centroid (ppm).

Parameters

- **path** – Target file path the <path>.imzML and the <path>.ibd is written to.
- **centroids** – a list of centroids.

Example usage:

```
import m2aia as m2

I = m2.ImzMLReader("path/to/imzMl/file.imzML")
I.Execute()
I.SetTolerance(75)
I.WriteContinuousCentroidImzML("path/to/imzMl/file.imzML", [300, 400, 500])
```

dir() → Path

Absolute path to directory containing the referenced imzML

name() → str

Name (including file ending) of the given imzML

path() → Path

Absolute path to the referenced imzML

CHAPTER

THREE

M2AIA.LIBRARY MODULE

m2aia.Library.**get_library**()

m2aia.Library.**get_shared_lib_dependencies**(*so_file_path*)

m2aia.Library.**load_library_dependencies_recursively**(*search_path: Path*, *library_name: str*, *dependencies: List*)

Load required M2aia libraries recursively

m2aia.Library.**load_m2aia_library**()

CHAPTER

FOUR

MODULE CONTENTS

m2aia.**prepare_environment**()

m2aia.**validate_environment**()

PYTHON MODULE INDEX

m

INDEX