sk-mad

a scikit for the estimation of missing audio data

Valentin Emiya et Ronan Hamon

Laboratoire d'Informatique Fondamentale (LIF)
(équipe Qarma)

14 décembre 2016

Outline

Outline

Motivations

- Collaborative research
  - Benefit from common tools
  - Benefit from other members' work
  - Work together
  - Benefit from other existing tools: sk-learn (machine learning), ltfatpy (time-frequency), librosa (audio processing)
- Reliable code/results
  - Reproducibility
  - Sharing code implies finalizing, formatting, refactoring, documenting
  - Unit tests
  - Integration and reuse make hidden bugs emerge
- Diffusion and promotion

Motivations

- Collaborative research
  - Benefit from common tools
  - Benefit from other members' work
  - Work together
  - Benefit from other existing tools: sk-learn (machine learning), ltfatpy (time-frequency), librosa (audio processing)
- Reliable code/results
  - Reproducibility
  - Sharing code implies finalizing, formatting, refactoring, documenting
  - Unit tests
  - Integration and reuse make hidden bugs emerge
- Diffusion and promotion

Motivations

- Collaborative research
  - Benefit from common tools
  - Benefit from other members' work
  - Work together
  - Benefit from other existing tools: sk-learn (machine learning), ltfatpy (time-frequency), librosa (audio processing)
- Reliable code/results
  - Reproducibility
  - Sharing code implies finalizing, formatting, refactoring, documenting
  - Unit tests
  - Integration and reuse make hidden bugs emerge
- Diffusion and promotion

Targetted usage

- **Development**
  - Defining common framework and architecture (do not reinvent it!)
  - Coding different blocks:
    - API to access datasets and load data
    - typical problems (declipping, loss of samples, time-frequency holes, etc.)
    - methods (nonnegative matrix factorization, phase reconstruction, etc.)
    - performance measure
- **Experimentation**
  - An standard experiment pattern (easy to design, to deploy, to share):
    - a given set of data items, e.g., from an available dataset
    - a given set of problem parameters to test
    - a given set of methods parameters to apply
    - storage and display of results
  - state-of-the-art solvers in addition to new ones
  - do not recode the full experimental chain
- **Demonstration**
  - experiments for a specific publication
  - examples to promote the research

Targetted usage

- **Development**
  - Defining common framework and architecture (do not reinvent it!)
  - Coding different blocks:
    - API to access datasets and load data
    - typical problems (declipping, loss of samples, time-frequency holes, etc.)
    - methods (nonnegative matrix factorization, phase reconstruction, etc.)
    - performance measure
- **Experimentation**
  - An standard experiment pattern (easy to design, to deploy, to share):
    - a given set of data items, e.g., from an available dataset
    - a given set of problem parameters to test
    - a given set of methods parameters to apply
    - storage and display of results
  - state-of-the-art solvers in addition to new ones
  - do not recode the full experimental chain
- **Demonstration**
  - experiments for a specific publication
  - examples to promote the research

Targetted usage

- **Development**
  - Defining common framework and architecture (do not reinvent it!)
  - Coding different blocks:
    - API to access datasets and load data
    - typical problems (declipping, loss of samples, time-frequency holes, etc.)
    - methods (nonnegative matrix factorization, phase reconstruction, etc.)
    - performance measure
- **Experimentation**
  - An standard experiment pattern (easy to design, to deploy, to share):
    - a given set of data items, e.g., from an available dataset
    - a given set of problem parameters to test
    - a given set of methods parameters to apply
    - storage and display of results
  - state-of-the-art solvers in addition to new ones
  - do not recode the full experimental chain
- **Demonstration**
  - experiments for a specific publication
  - examples to promote the research

Contributors and roles

- Ronan and Valentin: framework design, main effort
- Florent and Denis: councelling, support
- members of project MAD: first, use it; possibly, contribute
- Others: later

Roadmap

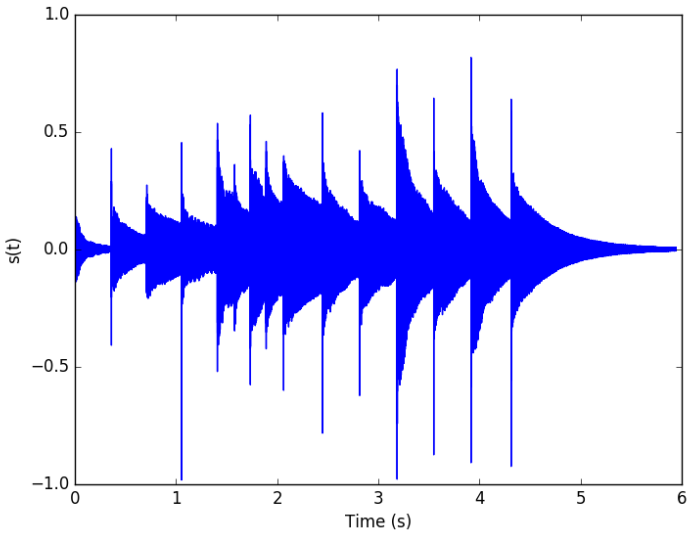
- January-December 2015: `ltfatpy` by Denis and Florent
- Decembre 2015 : public release of `ltfatpy`
- November 2015 - today : large effort by Ronan and Valentin to set up the `sk-mad` framework
- Today
  - Architecture available
  - For each type of components, at least one example module
  - Unit tests available
  - Documentation available
  - You can start using it when you want (not a first major release)
- Tomorrow : feed skmad with new modules for inpainting, develop new features, first release,...

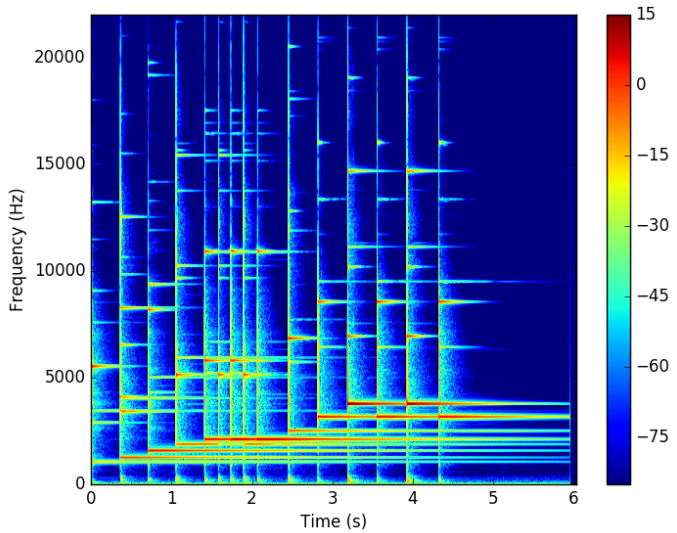
Outline

Example 1: spectrogram

```
>>> from skmad.data.datasets.ltfat.base import LtfatAudioFile
>>> from skmad.time_frequency.stft import Stft
>>> from skmad.measures.performance import sdr

>>> x = LtfatAudioFile.waveform_from_id('gspi')
>>> print(x)
Waveform, fs=44100Hz, length=262144, 0.0% of missing data
>>> my_stft = Stft(window='hann', win_len=2048, hop=256, nb_bins=4096)
>>> my_istft = my_stft.get_istft()
>>> X = my_stft.transform(x)
>>> y = my_istft.transform(X)
>>> print('Reconstruction error (SDR): {} dB'.format(sdr(x, y)))
Reconstruction error (SDR): 314.76 dB
>>> plt.figure() ; x.plot() ; X.plot()
```





Example 2: Experiments (1/2)

```
from skmad.experiments.base import Experiment

class MLSP2016_SyntheticDataNmf(Experiment):
    [...]
    def get_data(self, data_params):

        np.random.seed(data_params['seed'])
        shape = data_params['shape']
        V = np.random.random(shape)

        return {'V': V}

    def get_problem(self, problem_params):

        width = problem_params['width']
        mask = rectangular_mask(V.shape, width=width, height=0.4)

        return OcclusionProblem(mask=mask)

    def get_solver(self, solver_params):

        Solver = soler_params['Solver']
        return Solver(n_components=solver_params['K'], beta=solver_params['beta'])

    [...]
```

Example 2: Experiments (2/2)

```
data_params = {'shape':[(10, 20), (100, 200), (1000, 2000)]}
problem_params = {'width':[0.3, 0.4, 0.5]}
solver_params = {Solver:[SolverNmf, SolverCnmf]}


xp = MLSP2016_SyntheticDataNmf([...])
xp.add_tasks(data_params, problem_params, solver_params)
xp.run_task(10)
xp.launch_experiment(oar=True)
xp.collect_results()
```

Outline

Core modules

- **Datasets** (`skmad.data`):
  - Generation of synthetic data
  - API to access datasets based on `sqlalchemy` (example sounds, SISEC, MAPS, PEASS)
- **Problems** (`skmad.problems`):
  - set problem parameters (number of missing data, clipping level, etc.)
  - problem input: raw data object (from `data`)
  - problem output: problem data to be processed + solution
  - examples: clipped waveform, large holes in a waveform, STFT with random missing coefficients, feature-informed STFT inpainting
- **Solvers** (`skmad.solvers`)
  - one or several combined blocks to solve problems
  - get a problem and return the estimated solution
- **Experiments** (`skmad.experiments`):
  - combine data+problems+solvers
  - run (on cluster), save and plot results

Core modules

- **Datasets** (`skmad.data`):
  - Generation of synthetic data
  - API to access datasets based on `sqlalchemy` (example sounds, SISEC, MAPS, PEASS)
- **Problems** (`skmad.problems`):
  - set problem parameters (number of missing data, clipping level, etc.)
  - problem input: raw data object (from `data`)
  - problem output: problem data to be processed + solution
  - examples: clipped waveform, large holes in a waveform, STFT with random missing coefficients, feature-informed STFT inpainting
- **Solvers** (`skmad.solvers`)
  - one or several combined blocks to solve problems
  - get a problem and return the estimated solution
- **Experiments** (`skmad.experiments`):
  - combine data+problems+solvers
  - run (on cluster), save and plot results

Core modules

- **Datasets** (`skmad.data`):
  - Generation of synthetic data
  - API to access datasets based on `sqlalchemy` (example sounds, SISEC, MAPS, PEASS)
- **Problems** (`skmad.problems`):
  - set problem parameters (number of missing data, clipping level, etc.)
  - problem input: raw data object (from `data`)
  - problem output: problem data to be processed + solution
  - examples: clipped waveform, large holes in a waveform, STFT with random missing coefficients, feature-informed STFT inpainting
- **Solvers** (`skmad.solvers`)
  - one or several combined blocks to solve problems
  - get a problem and return the estimated solution
- **Experiments** (`skmad.experiments`):
  - combine data+problems+solvers
  - run (on cluster), save and plot results

Core modules

- **Datasets** (`skmad.data`):
  - Generation of synthetic data
  - API to access datasets based on `sqlalchemy` (example sounds, SISEC, MAPS, PEASS)
- **Problems** (`skmad.problems`):
  - set problem parameters (number of missing data, clipping level, etc.)
  - problem input: raw data object (from `data`)
  - problem output: problem data to be processed + solution
  - examples: clipped waveform, large holes in a waveform, STFT with random missing coefficients, feature-informed STFT inpainting
- **Solvers** (`skmad.solvers`)
  - one or several combined blocks to solve problems
  - get a problem and return the estimated solution
- **Experiments** (`skmad.experiments`):
  - combine data+problems+solvers
  - run (on cluster), save and plot results

Utils modules

- **Data structures**
  - Waveform / StftData / LabelledArray (masked array)
  - attributes for data, parameters (sampling freq.)
  - methods to create, plot, import/export
  - special class for masked data
- **Processing blocks**
  - block input/output: data objects
  - main method: `transform` (or fit_transform)
  - learning with `fit` method
  - example: feature extract., STFT, phase recons., mask generation, NMF
- **Documentation:**
  - Using `sphinx` module
  - inline documentation
- **Tests and coverage:**
  - $\sim$ 80% coverage

Utils modules

- **Data structures**
  - Waveform / StftData / LabelledArray (masked array)
  - attributes for data, parameters (sampling freq.)
  - methods to create, plot, import/export
  - special class for masked data
- **Processing blocks**
  - block input/output: data objects
  - main method: `transform` (or fit_transform)
  - learning with `fit` method
  - example: feature extract., STFT, phase recons., mask generation, NMF
- **Documentation:**
  - Using `sphinx` module
  - inline documentation
- **Tests and coverage:**
  - $\sim$ 80% coverage

Utils modules

- **Data structures**
  - Waveform / StftData / LabelledArray (masked array)
  - attributes for data, parameters (sampling freq.)
  - methods to create, plot, import/export
  - special class for masked data
- **Processing blocks**
  - block input/output: data objects
  - main method: `transform` (or fit_transform)
  - learning with `fit` method
  - example: feature extract., STFT, phase recons., mask generation, NMF
- **Documentation:**
  - Using `sphinx` module
  - inline documentation
- **Tests and coverage:**
  - $\sim$ 80% coverage

Utils modules

- **Data structures**
  - Waveform / StftData / LabelledArray (masked array)
  - attributes for data, parameters (sampling freq.)
  - methods to create, plot, import/export
  - special class for masked data
- **Processing blocks**
  - block input/output: data objects
  - main method: `transform` (or fit_transform)
  - learning with `fit` method
  - example: feature extract., STFT, phase recons., mask generation, NMF
- **Documentation:**
  - Using `sphinx` module
  - inline documentation
- **Tests and coverage:**
  - $\sim$ 80% coverage

Outline

Feedback

What is good

- Make different tools (datasets, nmf, time-frequency, etc...) accessible in Python
- Promotion of Python as a scientific language
- Time-saving in widely used actions: loading sound, stft, etc.

What is less good

- Time-consuming work for several reasons:
  - we are not experienced Python developers
  - ill-defined methods / objects
  - good intentions: unit tests, documentation
- Hard to focus on research and implementation

More generally

- Benefits in the long term
- Significant cost to start the project
- Indispensable component for reproducible research