Toolbox for Multimodal Learn (scikit-multimodallearn)
Dominique Benielli, Baptiste Bauvin, Cécile Capponi, Sokol Koço, Hachem Kadri, Riikka Huusari, François Laviolette

To cite this version:
Dominique Benielli, Baptiste Bauvin, Cécile Capponi, Sokol Koço, Hachem Kadri, et al.. Toolbox for Multimodal Learn (scikit-multimodallearn). 2021. hal-03473134

HAL Id: hal-03473134
https://hal.science/hal-03473134
Preprint submitted on 9 Dec 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Toolbox for Multimodal Learn

(scikit-multimodallearn)

Dominique Benielli, Baptiste Bauvin, Cécile Capponi, Sokol Koço, Hachem Kadri, Riikka Huusari, François Laviolette.

December 9, 2021

Abstract

scikit-multimodallearn is a Python library for multimodal supervised learning, licensed under Free BSD, and compatible with the well-known scikit-learn toolbox [Fabian Pedregosa, 2011]. This paper details the content of the library, including a specific multimodal data formatting and classification and regression algorithms. Use cases and examples are also provided.

1 Introduction

Learning from multiple views deals with the integration of different representations of the data, that can be either redundant, complementary, independent, or contradictory, in order to solve a learning problem [Cesa-Bianchi et al., 2010, Sun, 2013]. Thus, learning over all the views is expected to produce a final model that performs better than models learnt on individual views separately. Multi-view learning is a setting of machine learning which was initially devoted to semi-supervised tasks with view agreement [Blum and Mitchell, 1998], and later approaches penalizing view disagreements [Janodet et al., 2009], while mainly considering only two views. Surfing along the supervised side of multi-view learning, the assumption on view agreement is no longer considered, for the focus is set on exploring the complementarities (or diversity) of views. Methods of supervised multi-view learning include Multiple Kernel Learning [Bach et al., 2004, Gonen and Alpaydin, 2011], multi-view machines [Cao et al., 2016], ensemble-based methods [Koço and Capponi, 2011, Goyal et al., 2017], etc.

This paper presents a collaborative toolbox which implements algorithms for multimodal supervised learning where each labelled example is observed
under several views. Compatible with scikit-learn and numpy libraries, it comes with a intuitive input format for multi-view data sets which allows the use embedded algorithms in model selection facilities of sklearn. In this paper, we consider that multi-view and multi-modal can be used interchangeably to qualify the type of data set we present, even if it can have different meaning in some fields.

2 Toolbox scikit-multimodallearn at a Glance

The toolbox scikit-multimodallearn aims at providing a unified format for multi-view data representation by implementing a data structure in the form on a python class inheriting from the numpy array class. Empowered by this novel data structure, we propose the implementation of four multi-view algorithms as python classes each implementing a classifier and rigorously respecting the scikit-learn syntax for learning algorithms. A detailed documentation and a technical manual are available online.

2.1 Installing and Discovering scikit-multimodallearn

The installation of the toolbox is made easy by the use of pip: one just has to execute pip install scikit-multimodallearn in a terminal, or pip install -e . if you have sources in local. If needed, the package can be downloaded from Github. The development has been performed using continuous integration with Docker and automated tests, covering 90% of the code. A technical documentation, including the automated Sphinx documentation and examples are available online to allow collaborative development.

2.2 Input Data Format

Multi-view data are encoded by MultiModalArray, a python class which encapsulates information in numpy array-like format adapted for multi-modal learning. A first attempt to deal with sparse data representation is also included in the MultiModalSparseArray class.

This groundwork is crucial for multi-view algorithm development as it allows to implement algorithms that can treat multi-view data sets while

---

1Hosted here [https://dev.pages.lis-lab.fr/scikit-multimodallearn/](https://dev.pages.lis-lab.fr/scikit-multimodallearn/).
2Hosted here [https://github.com/dbenielli/scikit-multimodallearn](https://github.com/dbenielli/scikit-multimodallearn).
being compatible with the very useful `scikit-learn` library. In particular, algorithms implemented with `scikit-multimodallearn` take the exact same arguments as their `scikit-learn` basic counterparts: `fit`, `predict` and `score`, take either a `MultiModalArray`, `MultiModalSparseArray`, a dictionary or a simple `numpy.array` coupled with a list of view indices range in `views_ind` as input, enabling `scikit-multimodallearn` to be as versatile and user-friendly as possible. In the following example, a multiview data set is instantiated with a multi-view version of MNist [Deng, 2012] for which we generated the HOG [Triggs and Dalal, 2005] in 12 directions, and selected 3 random directions for each view.

```python
>>> import numpy as np
>>> from multimodal.datasets.data_sample import MultiModalArray
>>> from multimodal.datasets.base import load_dict
>>> from multimodal.tests.data.get_dataset_path import get_dataset_path
>>> dic_digit = load_dict(get_dataset_path("multiview_mnist.npy"))
>>> y = np.load(get_dataset_path("mnist_y.npy"))
>>> XX = MultiModalArray(dic_digit)
>>> XX.shape
(5000, 768)
>>> XX.n_views
4
>>> XX.views_ind
array([  0, 192, 384, 576, 768])
```

### 2.3 Architecture: Classifier Class and Algorithms

As a second major contribution, this library provides four multi-view classification algorithms that are designed to learn with multiple views (i.e. not restricted to only two views). These algorithms fall into two categories: kernel-based binary classifiers such as MVML [Huusari et al., 2018], a classification version of the MKL algorithm [Kloft et al., 2011]; and multi-class boosting algorithms such as MumBo [Koço and Capponi, 2011] and Mu-Combo [Koço, 2013]:

- **MVML** uses machinery from the theory of operator-valued kernels and vector-valued RKHS to learn simultaneously a vector-valued function to solve the supervised learning problem, and a metric acting between each pair of views in the data.

- **lp-MKL** provides co-regularization and multiple kernel learning which
are two well known kernel-based frameworks for learning in the presence of multiple views of data.

- **Mumbo** encourages the collaboration between major and minor views, in order to enhance the performances of classifiers usually learnt only on the major view.

- **MuComBo** is adapted to imbalanced learning problems, its adds a balanced weight on classes to the collaboration between view of Mumbo.

The algorithms are implemented in four main classes: `MVML`, `MKL`, `MumboClassifier`, `MuComboClassifier`, and two parent abstract meta classes `MKernel` and `UBoosting` for code factorization. As required by `scikit-learn`, the main class inherits from `BaseEstimator` or `BaseEnsemble` and `ClassifierMixin` from `sklearn.base`.

In addition, the kernel-based algorithms scale up to large training sets using a block-wise Nyström approximation of the multi-view kernel matrix (proposed as option).

Finally, as the code for these algorithms respects the guidelines given by `scikit-learn`, it is easy to add multi-view algorithms to the library using the tools provided by `scikit-learn` to test the compatibility and the data format provided by `scikit-multimodallearn`.

### 2.4 Multimodal Estimators Use Case

We present in the following a use case for initializing a `MumboClassifier` estimator.

```
>>> # import
>>> from multimodal.boosting.mumbo import MumboClassifier
>>> from sklearn.tree import DecisionTreeClassifier
>>> from sklearn.model_selection import train_test_split, cross_val_score
>>> # instantiation of weak classifier
>>> base_estimator = DecisionTreeClassifier(max_depth=4)
>>> # split data in train and test
>>> X_train, X_test, y_train, y_test = train_test_split(XX, y)
>>> # instantiation of MumboClassifier classifier
>>> clf = MumboClassifier(base_estimator=base_estimator,
                        n_estimators=4, random_state=7)
>>> # call fit method
>>> clf.fit(X_train, y=y_train)
>>> # call predict method
>>> y_pred = clf.predict(X_test))
```
MuComBoClassifier, MVML and MKL can be used the same way, and MuComBoClassifier with base_estimator parameter initialized to a (list of) monoview classifier(s), the default classifier being the default DecisionTreeClassifier of scikit-learn.

3 scikit-learn Compatibility

A crucial advantage stemming from the compatibility of scikit-multimodallearn with scikit-learn is the underlying inheritance of several functionalities, such as:

- `train_test_split` of scikit-learn can be used with MultiModalArray as demonstrated in [2.4](#).
- `cross_val_score` can be used transparently on multimodal estimator for multiviews data.

```python
>>> # Instantiating MuComboClassifier
>>> clfm = MuComboClassifier(base_estimator=base_estimator)
>>> from sklearn.model_selection import cross_val_score
>>> # usage cross_val_score on MuComboClassifier
>>> cross_val_score(clfm, XX, y, cv=5)
array([0.96666667, 0.96666667, 0.9, 0.93333333, 1.])
```

- `grid_search`, can also be used directly, for example, through OneVsOneClassifier for the non-multiclass MVML.

```python
>>> # import section
>>> from sklearn.multiclass import OneVsOneClassifier
>>> from sklearn.model_selection import GridSearchCV
>>> from multimodal.kernels.mvm import MVML
>>> # Instantiating a one versus one classifier based on MVML
>>> est = OneVsOneClassifier(MVML(lmbda=0.1, eta=1,
                                 nystrom_param=0.2))
>>> # Defining the hyper-parameter grid
>>> param = {'estimator__learn_A': (1, 3), 'estimator__learn_w':}
>>>            : (0, 1),
```
Figure 1: Accuracy results on a multi-view version of the multi-class MNist data set, computed with SuMMIT [Bauvin et al., 2021] in which early and late fusion are already implemented.

```python
'estimator__n_loops': (6, 10), 'estimator__nystrom_param':
(1.0, 0.3),
'estimator__kernel': ('linear', 'polynomial'),
'estimator__lmbda': (0.1,), 'estimator__eta': (1,)}

>>> # Instantiating the grid search object and fitting it
>>> grid = GridSearchCV(est, param, cv=5).fit(XX, y)
>>> # Getting the best score
>>> grid.best_score_
0.9693934335002783
>>> grid.best_params_
{'estimator__eta': 1, ... 'estimator__n_loops': 6, 'estimator__nystrom_param': 1.0}
```

4 Results

Figure 1 shows the results of a benchmark, with 5-folds cross validation on the multi-view version of MNist presented in Section 2.2. We tested a mono-view decision tree and Adaboost [Schapire, 2013] on each view, their early fusion versions and the four algorithms of scikit-multimodallearn. All the implemented algorithms output higher accuracy scores than the mono-view approaches and the naive fusion methods. MuCombo, despite being dedicated to imbalance problems, still displays an interesting score for this balanced task.
5 Conclusion

*scikit-multimodallearn* offers specialized algorithms dedicated to multi-modal supervised learning developed in Python3 on free licence, with an easy and transparent use for the user accustomed to the uses of *scikit-learn*. We are currently working on a more robust integration of sparse matrices and the addition of a data format allowing to load the views dynamically when needed, such as HDF5. Finally, the *scikit-learn* and SuMMIT compatibility will allow the multi-view community to develop their own algorithms on this framework to build a large library.

Acknowledgements

This work is supported by National Science and Engineering Research Council of Canada (NSERC) Discovery grant 262067, and granted by Lives Project (ANR-15-CE23-0026).

References


B. Bauvin, D. Benielli, C. Capponi, and F. Laviolette. Integrating and reporting full multi-view supervised learning experiments using SuMMIT. working paper or preprint, Apr. 2021. URL [https://hal.archives-ouvertes.fr/hal-03197079](https://hal.archives-ouvertes.fr/hal-03197079).


