mvlearn: Multiview Machine Learning in Python

Ronan Perry¹
Gavin Mischler⁹
Richard Guo²
Theodore Lee¹
Alexander Chang¹
Arman Koul¹
Cameron Franz²
Hugo Richard⁵
Iain Carmichael⁶
Pierre Ablin⁷
Alexandre Gramfort⁵
Joshua T. Vogelstein¹,3,4,8*

¹ Department of Biomedical Engineering, Johns Hopkins University, Baltimore, MD 21218
² Department of Computer Science, Johns Hopkins University, Baltimore, MD 21218
³ Center for Imaging Science, Johns Hopkins University, Baltimore, MD 21218
⁴ Kavli Neuroscience Discovery Institute, Institute for Computational Medicine, Johns Hopkins University, Baltimore, MD 21218
⁵ Université Paris-Saclay, Inria, Palaiseau, France
⁶ Department of Statistics, University of Washington, Seattle, WA 98195
⁷ CNRS and DMA, École Normale Supérieure, PSL University, Paris, France
⁸ Progressive Learning
⁹ Department of Electrical Engineering, Columbia University, New York, NY 10027
* Corresponding author

Editor: Joaquin Vanschoren

Abstract

As data are generated more and more from multiple disparate sources, multiview data sets, where each sample has features in distinct views, have grown in recent years. However, no comprehensive package exists that enables non-specialists to use these methods easily. mvlearn is a Python library which implements the leading multiview machine learning methods. Its simple API closely follows that of scikit-learn for increased ease-of-use. The package can be installed from Python Package Index (PyPI) and the conda package manager and is released under the MIT open-source license. The documentation, detailed examples, and all releases are available at https://mvlearn.github.io/

Keywords: multiview, machine learning, python, multi-modal, multi-table, multi-block

1. Introduction

Multiview data (sometimes referred to as multi-modal, multi-table, or multi-block data), in which each sample is represented by multiple views of distinct features, are often seen...
in real-world data and related methods have grown in popularity. A view is defined as a partition of the complete set of feature variables (Xu et al., 2013). Depending on the domain, these views may arise naturally from unique sources, or they may correspond to subsets of the same underlying feature space. For example, a doctor may have an MRI scan, a CT scan, and the answers to a clinical questionnaire for a diseased patient. However, classical methods for inference and analysis are often poorly suited to account for multiple views of the same sample, since they cannot properly account for complementing views that hold differing statistical properties (Zhao et al., 2017). To deal with this, many multiview learning methods have been developed to take advantage of multiple data views and produce better results in various tasks (Sun, 2013; Hardoon et al., 2004; Chao et al., 2017; Yang et al., 2014).

Although multiview learning techniques are increasingly seen in the literature, no open-source Python package exists which implements an extensive variety of methods. The most relevant existing package, multiview (Kanaan-Izquierdo et al., 2019), only includes 3 algorithms with an inconsistent API. mvlearn fills this gap with a wide range of well-documented algorithms that address multiview learning in different areas, including clustering, semi-supervised classification, supervised classification, and joint subspace learning. Additionally, mvlearn preprocessing tools can be used to generate multiple views from a single original data matrix, expanding the use-cases of multiview methods and potentially improving results over typical single-view methods with the same data (Sun, 2013). Subsampled sets of features have notably led to successful ensembles of independent single-view algorithms (Ho, 1998) but can also be taken advantage of jointly by multiview algorithms to reduce variance in unsupervised dimensionality reduction (Foster et al., 2008) and improve supervised task accuracy (Nigam and Ghani, 2000). The last column of Table 1 details which methods may be useful on single-view data after feature subsampling. mvlearn has been tested on Linux, Mac, and PC platforms, and adheres to strong code quality principles. Continuous integration ensures compatibility with past versions, PEP8 style guidelines keep the source code clean, and unit tests provide over 95% code coverage at the time of release.

2. API Design

The API closely follows that of scikit-learn (Pedregosa et al., 2011) to make the package accessible to those with even basic knowledge of machine learning in Python (Buitinck et al., 2013). The main object type in mvlearn is the estimator object, which is modeled after scikit-learn’s estimator. mvlearn changes the familiar method fit\((X, y)\) into a multiview equivalent, fit\((Xs, y)\), where \(Xs\) is a list of data matrices, corresponding to a set of views with matched samples (i.e. the \(i\)th row of each matrix represents the features of the same \(i\)th sample across views). Note that \(Xs\) need not be a third-order tensor as each view need not have the same number of features. As in scikit-learn, classes which make a prediction implement predict\((Xs)\), or fit.predict\((Xs, y)\) if the algorithm requires them to be performed jointly, where the labels \(y\) are only used in supervised algorithms. Similarly, all classes which transform views, such as all the embedding methods, implement transform\((Xs)\) or fit.transform\((Xs, y)\).
3. Library Overview

mvlearn includes a wide breadth of method categories and ensures that each offers enough depth so that users can select the algorithm that best suits their data. The package is organized into the modules listed below which includes the multiview algorithms in Table 1 as well as various utility and preprocessing functions. The modules’ summaries describe their use and fundamental application.

**Decomposition:** mvlearn implements the Angle-based Joint and Individual Variation Explained (AJIVE) algorithm (Feng et al., 2018), an updated version of the JIVE algorithm (Lock et al., 2013). This was originally developed to deal with genomic data and characterize similarities and differences between data sets. mvlearn also implements multiview independent component analysis (ICA) methods (Calhoun et al., 2001; Richard et al., 2020), originally developed for fMRI processing.

**Cluster:** mvlearn contains multiple algorithms for multiview clustering, which can better take advantage of multiview data by using unsupervised adaptations of co-training. Even when the only apparent distinction between views is the data type

---

<table>
<thead>
<tr>
<th>Module</th>
<th>Algorithm (Reference)</th>
<th>Maximum Views</th>
<th>Useful on Constructed Data from a Single Original View</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decomposition</td>
<td>AJIVE (Feng et al., 2018)</td>
<td>2</td>
<td>X</td>
</tr>
<tr>
<td>Decomposition</td>
<td>Group PCA/ICA (Calhoun et al., 2001)</td>
<td>≥ 2</td>
<td>X</td>
</tr>
<tr>
<td>Decomposition</td>
<td>Multiview ICA (Richard et al., 2020)</td>
<td>≥ 2</td>
<td>X</td>
</tr>
<tr>
<td>Cluster</td>
<td>MV K-Means (Bickel and Scheffer, 2004)</td>
<td>2</td>
<td>✓</td>
</tr>
<tr>
<td>Cluster</td>
<td>MV Spherical K-Means (Bickel and Scheffer, 2004)</td>
<td>2</td>
<td>✓</td>
</tr>
<tr>
<td>Cluster</td>
<td>MV Spectral Clustering (Kumar and Daume, 2011)</td>
<td>≥ 2</td>
<td>✓</td>
</tr>
<tr>
<td>Cluster</td>
<td>Co-regularized MV Spectral Clustering (Kumar et al., 2011)</td>
<td>≥ 2</td>
<td>✓</td>
</tr>
<tr>
<td>Semi-supervised</td>
<td>Co-training Classifier (Blum and Mitchell, 1998)</td>
<td>2</td>
<td>✓</td>
</tr>
<tr>
<td>Semi-supervised</td>
<td>Co-training Regressor (Zhou and Li, 2005)</td>
<td>2</td>
<td>✓</td>
</tr>
<tr>
<td>Embed</td>
<td>CCA (Hotelling, 1936)</td>
<td>2</td>
<td>X</td>
</tr>
<tr>
<td>Embed</td>
<td>Multi CCA (Tenenhaus and Tenenhaus, 2011)</td>
<td>≥ 2</td>
<td>X</td>
</tr>
<tr>
<td>Embed</td>
<td>Kernel Multi CCA (Hardoon et al., 2004)</td>
<td>≥ 2</td>
<td>X</td>
</tr>
<tr>
<td>Embed</td>
<td>Deep CCA (Andrew et al., 2013)</td>
<td>2</td>
<td>X</td>
</tr>
<tr>
<td>Embed</td>
<td>Generalized CCA (Afshin-Pour et al., 2012)</td>
<td>≥ 2</td>
<td>X</td>
</tr>
<tr>
<td>Embed</td>
<td>MV Multi-dimensional Scaling (MVMDS) (Trendafilov, 2010)</td>
<td>≥ 2</td>
<td>X</td>
</tr>
<tr>
<td>Embed</td>
<td>Omnibus Embed (Levin et al., 2017)</td>
<td>≥ 2</td>
<td>X</td>
</tr>
<tr>
<td>Embed</td>
<td>Split Autoencoder (Wang et al., 2015)</td>
<td>2</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 1: Multiview (MV) algorithms offered in mvlearn and their properties.
of certain features, such as categorical and continuous variables, multiview clustering has been very successful (Chao et al., 2017).

**Semi-supervised:** Semi-supervised classification (which includes fully-supervised classification as a special case) is implemented with the co-training framework (Blum and Mitchell, 1998), which uses information from complementary views of (potentially) partially labeled data to train a classification system. If desired, the user can specify nearly any type of classifier for each view, specifically any scikit-learn-compatible classifier which implements a `predict_proba` method. Additionally, the package offers semi-supervised regression (Zhou and Li, 2005) using the co-training framework.

**Embed:** mvlearn offers an extensive suite of algorithms for learning latent space embeddings and joint representations of views. One category is canonical correlation analysis (CCA) methods, which learn transformations of two views such that the outputs are highly correlated. Many software libraries include basic CCA, but mvlearn also implements several more general variants, including multiview CCA (Tenenhaus and Tenenhaus, 2011) for more than two views, Kernel multiview CCA (Hardoon et al., 2004; Bach and Jordan, 2003; Kuss and Graepel, 2003), Deep CCA (Andrew et al., 2013), and Generalized CCA (Afshin-Pour et al., 2012) which is efficiently parallelizable to any number of views. Several other methods for dimensionality reduction and joint subspace learning are included as well, such as multiview multidimensional scaling (Trendafilov, 2010), omnibus embedding (Levin et al., 2017), and a split autoencoder (Wang et al., 2015).

**Compose:** Several functions for integrating single-view and multiview methods are implemented, facilitating operations such as preprocessing, merging, or creating multiview data sets. If the user only has a single view of data, view-generation algorithms in this module such as Gaussian random projections and random subspace projections allow multiview methods to still be applied and may improve upon results from single-view methods (Sun, 2013; Nigam and Ghani, 2000; Ho, 1998).

**Data sets and Plotting:** A synthetic multiview data generator as well as dataloaders for the Multiple Features Data Set (Breukelen et al., 1998) in the UCI repository (Dua and Graff, 2017) and the genomics Nutrimouse data set (Martin et al., 2007) are included. Also, plotting tools extend matplotlib and seaborn to facilitate visualizing multiview data.

### 4. Conclusion

mvlearn introduces an extensive collection of multiview learning tools, enabling anyone to readily access and apply such methods to their data. As an open-source package, mvlearn welcomes contributors to add new desired functionality to further increase its applicability and appeal. As data are generated from more diverse sources and the use of machine learning extends to new fields, multiview learning techniques will be more useful to effectively extract information from real-world data sets. With these methods accessible to non-specialists, multiview learning algorithms will be able to improve results in academic and industry applications of machine learning.
Author Contribution


Acknowledgements

This work is supported by the Defense Advanced Research Projects Agency (DARPA) Lifelong Learning Machines program through contract FA8650-18-2-7834, by R01 AG066184/AG/NIA NIH HHS/United States, and through funding from Microsoft Research. We thank the NeuroData Design class and the NeuroData lab at Johns Hopkins University for support and guidance as well as the reviewers for their helpful feedback.

References


