mvlearn: Multiview Machine Learning in Python

Ronan Perry[1], Gavin Mischler[1], Richard Guo[2], Theodore Lee[1], Alexander Chang[1], Arman Koul[1], Cameron Franz[2], and Joshua T. Vogelstein[1,3,4] *

Abstract. As data are generated more and more from multiple disparate sources, multiview data sets, where each sample has features in distinct views, have ballooned 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) or the `conda` package manager and is released under the Apache 2.0 open-source license. The documentation, detailed tutorials, and all releases are available at https://mvlearn.neurodata.io/.

Key words. multiview, machine learning, python

1 Introduction Multiview 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 [1]. 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 [2]. 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 [3–6].

Although multiview learning techniques are increasingly utilized in literature, no open-source Python package exists which implements an extensive variety of methods. The most relevant existing package, `multiview` [7], 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` 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 this data [3, 8, 9]. `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 90% code coverage at the time of release. Table 1 summarizes the multiview algorithms implemented in `mvlearn`. The rest of this paper describes the API design, main features, and examples of using the package.

2 API Design The API closely follows that of `scikit-learn` [21] to make the package accessible to those with even basic knowledge of machine learning in Python [22]. 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). As in `scikit-learn` [21], 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,y)` or `fit_transform(Xs)`.

*Corresponding author: jovo@jhu.edu [1] Department of Biomedical Engineering Johns Hopkins University, [2] Department of Computer Science, Johns Hopkins University, [3] Center for Imaging Science, Institute for Computational Medicine, Kavli Neuroscience Discovery Institute, Johns Hopkins University, [4] Progressive Learning

Module	Algorithm [Reference]	Maximum Views	Useful on Constructed Data from a Single Original View
Decomposition	Angle-based Joint and Individual Variation Explained (AJIVE) [10]	2	✗
Decomposition	Multiview ICA [11]	≥ 2	✗
Cluster	MV K-Means [12]	2	✓
Cluster	MV Spherical K-Means [12]	2	✓
Cluster	MV Spectral Clustering [13]	≥ 2	✓
Cluster	Co-regularized MV Spectral Clustering [14]	≥ 2	✓
Semi-supervised	Co-training Classifier [15]	2	✓
Embed	Kernel CCA [4]	2	✗
Embed	Deep CCA [16]	2	✗
Embed	Generalized CCA [17]	≥ 2	✗
Embed	MV Multi-dimensional Scaling (MVMDS) [18]	≥ 2	✗
Embed	Omnibus Embed [19]	≥ 2	✗
Embed	Split Autoencoder [20]	2	✗

Table 1: **Multiview (MV) algorithms offered in `mvlearn` and their properties.**

3 Library Overview To build a package that is useful across the spectrum of applications for multiview learning, `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 [10], an updated version of the JIVE algorithm [23]. 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, developed for neural data.

Cluster: `mvlearn` contains multiple algorithms for multiview clustering, which can better take advantage of multiview data by using unsupervised adaptations of co-training [12–14]. Even when the only apparent distinction between views is the data type of certain features, such as categorical and continuous variables, multiview clustering has been very successful [5], making these methods widely applicable to real-world data.

Semi-supervised: Semi-supervised classification (which includes fully-supervised classification as a special case) is implemented with the co-training framework [15], 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.

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` implements several more general variants, including Kernel CCA [4] (with several pre-specified kernel options), Deep CCA [16], and Generalized CCA [17] 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 multi-dimensional scaling [18], omnibus embedding [19], and a split autoencoder [20], providing extensive functionality for learning common relationships between views of multiview data.

Construct: Even if the user only has a single view of data, view-generation algorithms can allow multiview methods to still be used effectively, and may improve results over single-view methods [3, 8, 9].

Data sets and Plotting: A synthetic multiview data generator as well as a dataloader for the Multiple Features Data Set [24] in the UCI repository [25] are included. Also, plotting tools extend `matplotlib` and `seaborn` to facilitate visualizing multiview data.

4 Code Examples Figure 1 provides an illustration of the performance of different CCA variants implemented in `mvlearn` when the two views hold different relationships. The data were simulated using the `mvlearn GaussianMixture` class. The first view was sampled from a two-dimensional Gaussian and the second view was constructed by applying a noisy transformation to the first view. To each view, two independent Gaussian noise dimensions were added. Three distinct transformations were applied (each row) and the CCA variants were run in each case (columns 2-5). Each plot in Figure 1 plots the first dimensions of the two views against one another, for both the raw views and the learned CCA variant transformations as applied to held-out data. Using the different CCA options in the package, the highly correlated (possibly nonlinear) latent relationship is uncovered in each case.

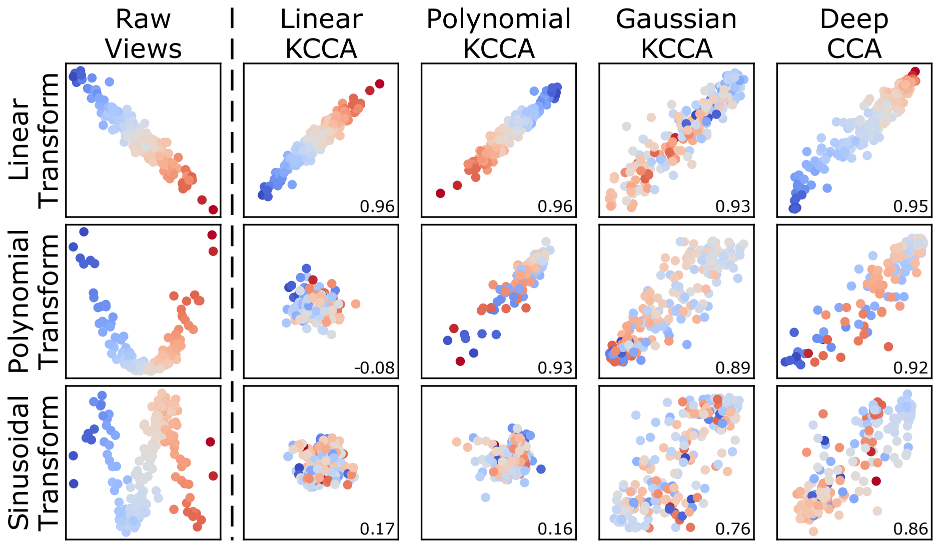

Figure 1: **Numerical illustration of data with two views (leftmost column) and embeddings using various algorithms in the `mvlearn` package.** Each plot in the first column shows the first dimension of view 1 vs. that of view 2, where view 2 is generated from a noisy transformation applied to view 1. Each row corresponds to view 2 data generated with the transformation listed on the y-axis. In columns 2-5, a different CCA variant is fit to each set of data and transforms held-out data. Plotted are the first dimensions of each transformed view, along with their Pearson correlation coefficient in the bottom right. Larger correlation indicates that the latent relationship is better uncovered. The color of a sample is consistent across rows, illustrating how the feature spaces are transformed.

Figure 2 demonstrates using `mvlearn` to estimate cluster assignments on six views of the Multiple Features Data Set [24] available within the package against four true underlying classes. For comparison, results from the naïve approach of concatenating the views into a single matrix are shown as well. Multiview dimensionality reduction produces better visual cluster separation in two dimensions than Principal Component Analysis (PCA) does, and multiview clustering achieves far superior clustering accuracy using the full feature space than classical spectral clustering.

```python
from mvlearn.datasets import load_UCImultifeature
from mvlearn.plotting import quick_visualize
from mvlearn.cluster import MultiviewSpectralClustering

# Load 4-class multiview data
Xs, y = load_UCImultifeature(select_labeled=[0,1,2,3])

# Use multiview multi-dimensional scaling to reduce 6 views to 2D
quick_visualize(Xs, labels=y, title="True Labels")

# Initialize multiview clustering object and estimate clusters
mv_clust = MultiviewSpectralClustering(n_clusters=4, random_state=42,
                                       affinity="nearest_neighbors")
mv_labels = mv_clust.fit_predict(Xs)

# Plot predicted cluster labels over the 2D visualization
quick_visualize(Xs, labels=mv_labels, title="Predicted Clusters")
```

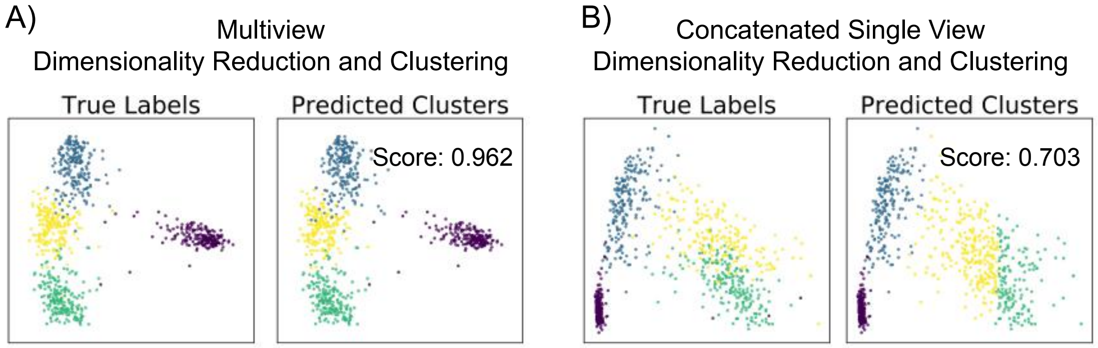

Figure 2: **Code snippet showing the use of `mvlearn` to analyze the Multiple Features Data Set.** This data set consists of six views of handwritten digit images, such as a view of Fourier coefficients, a view of morphological features, and others. Thus, in the code above, `Xs` is a list of six arrays. Although dimensionality reduction to 2D was performed for easy visualization, cluster assignments were estimated using all the features in all six views. A) 2D plots of the six views after reducing dimensionality with MVMDS. Points are colored by true (left) and predicted (right) clusters, achieving a homogeneity score of 0.962. B) Visualization of the six views after concatenating and reducing dimensionality with PCA (code not shown). Clusters predicted with traditional spectral clustering achieve a homogeneity score of 0.703. Full code for this demo is available in the tutorials of the documentation.

5 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.

6 Acknowledgements

This work is supported by the Defense Advanced Research Projects Agency (DARPA) Lifelong Learning Machines program through contract FA8650-18-2-7834 and through funding from Microsoft Research. We thank all the contributors for assisting with writing `mvlearn`. We thank the NeuroData Design class and the NeuroData lab at Johns Hopkins University for support and guidance.

References

[1] Chang Xu, Dacheng Tao, and Chao Xu. A survey on multi-view learning. *arXiv preprint arXiv:1304.5634*, 2013.

[2] Jing Zhao, Xijiong Xie, Xin Xu, and Shiliang Sun. Multi-view learning overview: Recent progress and new challenges. *Information Fusion*, 38:43–54, 2017.

[3] Shiliang Sun. A survey of multi-view machine learning. *Neural computing and applications*, 23 (7-8):2031–2038, 2013.

[4] David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 16(12):2639–2664, 2004.

[5] Guoqing Chao, Shiliang Sun, and Jinbo Bi. A survey on multi-view clustering. *arXiv preprint arXiv:1712.06246*, 2017.

[6] Yuhao Yang, Chao Lan, Xiaoli Li, Bo Luo, and Jun Huan. Automatic social circle detection using multi-view clustering. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1019–1028, 2014.

[7] Samir Kanaan-Izquierdo, Andrey Ziyatdinov, Maria Araceli Burgueño, and Alexandre Perera-Lluna. Multiview: a software package for multiview pattern recognition methods. *Bioinformatics*, 35(16): 2877–2879, 2019.

[8] Kamal Nigam and Rayid Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 86–93, 2000.

[9] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8):832–844, 1998.

[10] Qing Feng, Meilei Jiang, Jan Hannig, and JS Marron. Angle-based joint and individual variation explained. *Journal of multivariate analysis*, 166:241–265, 2018.

[11] Hugo Richard, Luigi Gresele, Aapo HyvÃď'rinen, Bertrand Thirion, Alexandre Gramfort, and Pierre Ablin. Modeling shared responses in neuroimaging studies through multiview ica, 2020.

[12] Steffen Bickel and Tobias Scheffer. Multi-view clustering. In *ICDM*, volume 4, pages 19–26, 2004.

[13] Abhishek Kumar and Hal Daumé. A co-training approach for multi-view spectral clustering. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 393–400, 2011.

[14] Abhishek Kumar, Piyush Rai, and Hal Daume. Co-regularized multi-view spectral clustering. In *Advances in neural information processing systems*, pages 1413–1421, 2011.

[15] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, 1998.

[16] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep canonical correlation analysis. In *International conference on machine learning*, pages 1247–1255, 2013.

[17] Babak Afshin-Pour, Gholam-Ali Hossein-Zadeh, Stephen C Strother, and Hamid Soltanian-Zadeh. Enhancing reproducibility of fmri statistical maps using generalized canonical correlation analysis in npairs framework. *NeuroImage*, 60(4):1970–1981, 2012.

[18] Nickolay T Trendafilov. Stepwise estimation of common principal components. *Computational Statistics & Data Analysis*, 54(12):3446–3457, 2010.

[19] Keith Levin, Avanti Athreya, Minh Tang, Vince Lyzinski, and Carey E Priebe. A central limit theorem for an omnibus embedding of multiple random dot product graphs. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 964–967. IEEE, 2017.

[20] Weiran Wang, Raman Arora, Karen Livescu, and Jeff Bilmes. On deep multi-view representation learning. In *International Conference on Machine Learning*, pages 1083–1092, 2015.

[21] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[22] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, et al. Api design for machine learning software: experiences from the scikit-learn project. *arXiv preprint arXiv:1309.0238*, 2013.

[23] Eric F Lock, Katherine A Hoadley, James Stephen Marron, and Andrew B Nobel. Joint and individual variation explained (jive) for integrated analysis of multiple data types. *The annals of applied statistics*, 7(1):523, 2013.

[24] Martijn van Breukelen, Robert PW Duin, David MJ Tax, and JE Den Hartog. Handwritten digit recognition by combined classifiers. *Kybernetika*, 34(4):381–386, 1998.

[25] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.