Combining Machine Learning Models Using `combo` Library

Yue Zhao,[1] **Xuejian Wang,**[1] **Cheng Cheng,**[1] **Xueying Ding**[2]
[1]H. John Heinz III College, Carnegie Mellon University, Pittsburgh, PA 15213 USA
[2]Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA 15213 USA
zhaoy@cmu.edu, {xuejianw, ccheng2, xding2}@andrew.cmu.edu

Abstract

Model combination, often regarded as a key sub-field of ensemble learning, has been widely used in both academic research and industry applications. To facilitate this process, we propose and implement an easy-to-use Python toolkit, `combo`, to aggregate models and scores under various scenarios, including classification, clustering, and anomaly detection. In a nutshell, `combo` provides a unified and consistent way to combine both raw and pretrained models from popular machine learning libraries, e.g., scikit-learn, XGBoost, and LightGBM. With accessibility and robustness in mind, `combo` is designed with detailed documentation, interactive examples, continuous integration, code coverage, and maintainability check; it can be installed easily through Python Package Index (PyPI) or https://github.com/yzhao062/combo.

Introduction

Recently, model combination has gained much attention in many real-world tasks, and stayed as the winning solution in numerous data science competitions such like Kaggle (2007). It is considered as a sub-field of ensemble learning, aiming for achieving better prediction performance (2012). Despite that, it is often beyond the scope of machine learning—it has been used in other domains such as the experimental design in clinical trials. Generally speaking, model combination has two key usages: stability improvement and performance boost. For instance, practitioners run independent trials and then average the results to eliminate the built-in randomness and uncertainty—more reliable results may be obtained. Additionally, even in a non-ideal scenario, base models may make independent but complementary errors. The combined model can, therefore, yield better performance than any constituent ones.

Although model combination is crucial for all sorts of learning tasks, dedicated Python libraries are absent. There are a few packages that partly fulfill this purpose, but established libraries either exist as single purpose tools like PyOD (2019) and pycobra (2018), or as part of general purpose libraries like scikit-learn (2011).

```
>>> from combo.models.classifier_dcs
    import DCS
    # initialize a group of classifiers
>>> classifiers = [
      DecisionTreeClassifier(),
      LogisticRegression(),
      KNeighborsClassifier()]
>>> # initialize/fit the combination model
>>> clf = DCS(base_estimators=classifiers)
>>> clf.fit(X_train)
>>> # fit and make prediction
>>> y_test_pred = clf.predict(X_test)
>>> y_test_proba =
      clf.predict_proba(X_test)
>>> # fit and predict on the same dataset
>>> y_train_pred = clf.fit_predict(X_train)
```

Code Snippet 1: Demo of `combo` API with DCS

`combo` can fill this gap with four key advantages. Firstly, `combo` contains more than 15 combination algorithms, including both classical algorithms like dynamic classifier selection (DCS) (1997) and recent advancement like LCSP (2019). It could handle the combination operation for all sorts of tasks like classification, clustering, and anomaly detection. Secondly, `combo` works with both raw and pretrained learning models from major libraries like scikit-learn, XGBoost, and LightGBM, given certain conditions are met. Thirdly, the models in `combo` are designed with unified APIs, detailed documentation[1], and interactive examples[2] for the easy use. Lastly, all `combo` models are associated with unit test and being checked by continuous integration tools for robustness; code coverage and maintainability check are also enabled for performance and sustainability. To our best knowledge, this is the first comprehensive framework for combining learning models and scores in Python, which is valuable for data practitioners, machine learning researchers, and data competition participants.

[1]https://pycombo.readthedocs.io
[2]https://mybinder.org/v2/gh/yzhao062/combo/master

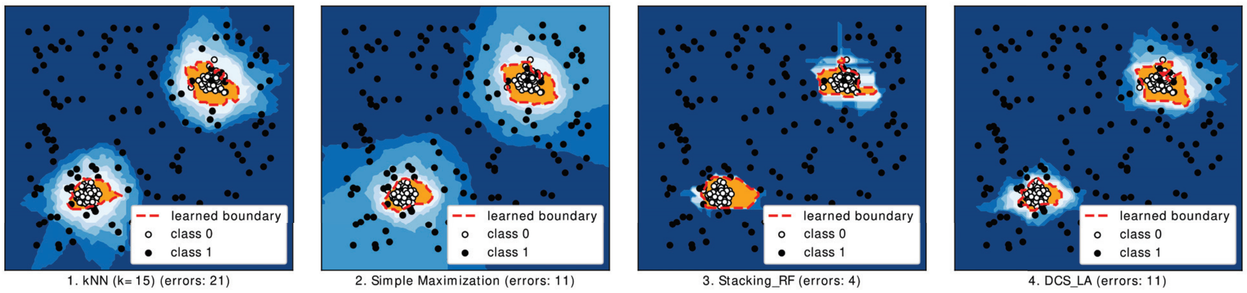

Figure 1: Comparison of Selected Classifier Combination on Simulated Data

Core Scenarios

`combo` models for classification, clustering, and anomaly detection share unified APIs. Inspired by scikit-learn's API design, the models in `combo` all come with the following key methods: (i) `fit` function processes the train data and gets the model ready for prediction; (ii) `predict` function generates labels for the unknown test data once the model is fitted; (iii) `predict_proba` generates predictions in probability instead of discrete labels by `predict` and (iv) `fit_predict` calls `fit` function first on the input data and then predicts on it (applicable to unsupervised model only). Code Snippet 1 shows the use of above APIs on DCS. Notably, fitted (pretrained) models can be used directly by setting `pre_fitted` flag; `fit` process will be skipped.

Classifier Combination aims to aggregate multiple base supervised classifiers in either parallel or sequential manner. Selected classifier combination methods implemented in `combo` include stacking (meta-learning), dynamic classifier selection, dynamic ensemble selection, and a group of heuristic aggregation methods like averaging and majority vote. Fig. 1 shows how different frameworks behave on a simulated dataset with 300 points. The leftmost one is a simple kNN model (k=15), and the other three are the combination of five kNN models with k in range $[5, 10, 15, 20, 25]$. Different from classifier combination, ***Cluster Combination*** is usually done in an unsupervised manner. The focus is on how to align the predicted labels generated by base clusterings, as cluster labels are categorical instead of ordinal. For instance, $[0, 1, 1, 0, 2]$ and $[1, 0, 0, 1, 2]$ are equivalent with appropriate alignment. Two classical clustering combination methods are therefore implemented to handle this—clustering combination using evidence accumulation (EAC)(2005) and Clusterer Ensemble (2006). ***Anomaly Detection*** concentrates on identifying the anomalous objects from the general data distribution (2019). The challenges of combining multiple outlier detectors lie in its unsupervised nature and extreme data imbalance. Two latest combination frameworks, LSCP (2019) and XGBOD (2018), are included in `combo` for unsupervised and semi-supervised detector combination. ***Score Combination*** comes with more flexibility than the above tasks as it only asks for the output from multiple models, whichever it is from a group of classifiers or outlier detectors. As a general purpose task, score combination methods are easy to use without the need of initializing a dedicated class. Each aggregation method, e.g., average of maximum (AOM), can be invoked directly.

Conclusion and Future Directions

`combo` is a comprehensive Python library to combine the models from major machine learning libraries. It supports four types of combination scenarios (classification, clustering, anomaly detection, and raw scores) with unified APIs, detailed documentation, and interactive examples. As avenues for future work, we will add the combination frameworks for customized deep learning models (from TensorFlow, PyTorch, and MXNet), enable GPU acceleration and parallelization for scalability, and expand to more task scenarios such as imbalanced learning and regression.

References

Bell, R. M., and Koren, Y. 2007. Lessons from the netflix prize challenge. *SIGGKDD Explorations* 9(2):75–79.

Fred, A. L., and Jain, A. K. 2005. Combining multiple clusterings using evidence accumulation. *PAMI* 27(6).

Guedj, B., and Desikan, B. S. 2018. Pycobra: A python toolbox for ensemble learning and visualisation. *JMLR* 18(190).

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. 2011. Scikit-learn: Machine learning in python. *JMLR* 12(Oct):2825–2830.

Woods, K.; Kegelmeyer, W. P.; and Bowyer, K. 1997. Combination of multiple classifiers using local accuracy estimates. *PAMI* 19(4):405–410.

Zhao, Y., and Hryniewicki, M. K. 2018. XGBOD: Improving Supervised Outlier Detection with Unsupervised Representation Learning. *IJCNN*.

Zhao, Y.; Nasrullah, Z.; Hryniewicki, M. K.; and Li, Z. 2019. LSCP: Locally selective combination in parallel outlier ensembles. In *SDM*, 585–593. SIAM.

Zhao, Y.; Nasrullah, Z.; and Li, Z. 2019. PyOD: A python toolbox for scalable outlier detection. *JMLR* 20(96):1–7.

Zhou, Z.-H., and Tang, W. 2006. Clusterer ensemble. *KBS* 19(1):77–83.

Zhou, Z.-H. 2012. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC.