scikit-activeml: A Library and Toolbox for Active Learning Algorithms

Daniel Kottke¹[0000−0002−7870−6033], Marek Herde¹[0000−0003−4908−122X], Tuan Pham Minh¹[0000−0002−5102−5561], Alexander Benz¹, Pascal Mergard¹, Atal Roghman¹, Christoph Sandrock¹, and Bernhard Sick¹[0000−0001−9467−656X]

¹University of Kassel
Wilhelmshöher Allee 73, 34121 Kassel, Germany
{daniel.kottke, marek.herde, tuan.pham, bsick}@uni-kassel.de

This document describes the main features and goals of scikit-activeml. Check out the repository at https://github.com/scikit-activeml/scikit-activeml

1 Introduction

Machine learning applications often need large amounts of training data to perform well. Whereas unlabeled data can be easily gathered, the labeling process is difficult, time-consuming, or expensive in most applications. Active learning [Settles, 2012] can help solve this problem by querying labels for those data points that will improve the performance the most. Thereby, the goal is that the learning algorithm performs sufficiently well with fewer labels.

We provide a library called scikit-activeml that covers the most relevant query strategies and implements tools to work with partially labeled data. It is programmed in Python and builds on top of scikit-learn [Pedregosa et al., 2011]. It is designed as a community project and was initialized in 2020 at the Intelligent Embedded Systems Group at University Kassel. During its development, we focus on the following:

− We build upon scikit-learn to use established design decisions that simplify getting started with our library for many users.
− We provide a modular structure and lots of useful functions such that researchers can easily implement and distribute their active learning strategies.
− We are application-oriented to help practitioners when realizing their machine learning applications.
− We aim at unifying algorithms and concepts to make strategies in the field of active learning comparable.
− This framework is an Open Source project and encourages others to join the development.

In the following, we describe this library’s general structure, provide an exemplary active learning cycle, and discuss future work and algorithms that we will include later.
2 Structural Overview

Our active learning library scikit-activeml adopts the design principles of the well-known machine learning framework scikit-learn. This also applies to the underlying code, which uses the two Python libraries numpy and scipy. As a result, our library’s functions and objects work internally with numpy arrays and scipy functions for efficient computation.

The central part of our library is a QueryStrategy. It is an interface inheriting from the BaseEstimator interface of scikit-learn. A class implementing this interface is enforced to provide the query method. This method is the core of a query strategy and responsible for selecting data points. The type of queries and their selection depends on the active learning paradigm. Therefore, there are further subinterfaces of QueryStrategy specifying the input and output of the query method regarding the respective active learning paradigm. For example, the interface for pool-based active learning strategies with a single annotator expects an array of candidate samples as input. One or multiple of these samples are selected for querying their labels.

Machine learning models represent another major part of our library next to query strategies. Currently, only classification models are supported. All of them implement the ClassifierMixin interface to ensure compatibility to the scikit-learn framework. The main difference to scikit-learn models is our extension towards missing labels. This way, each classification model can handle data samples whose class labels are unknown. Moreover, classification models with probabilistic outputs are equipped with cost-sensitive predictions. The use of scikit-learn classification models within our library is ensured by providing the additional functionalities through an easy-to-use wrapper class.

An abstract visualization of our library’s structure is presented in Fig. 1.

---

Fig. 1. Structure of the main parts of our library scikit-activeml. Each node represents a class or interface. The arrows illustrate the inheritance hierarchy among them. The functionality of a dashed node is not yet available in our library.
3 Exemplary Active Learning Cycle and Methods

Query strategies are usually mentioned in the context of an active learning application which is often visualized in the so-called active learning cycle. We show an exemplary implementation of such a cycle in Algorithm 1.

In an active learning application, we are usually faced with some unlabeled data $X$ that needs to be categorized into classes (here: 0 vs. 1). In our example, we create an unlabeled data-set using the `make_classification` function from `sklearn` (see line 8). Additionally, we also get the true labels $y_{true}$ from that function which are usually not available but can be requested from the oracle. To store the labels we acquired from the oracle, we introduce the vector $y$ that contains the labels we have so far. Initially, $y$ indicates that the labels for all instances are missing (MISSING_LABEL).

There are many easy-to-use classification algorithms in `sklearn`. For example, we use the logistic regression classifier in line 10. As `sklearn` classifiers cannot cope with missing labels, we need to wrap these with the `SklearnClassifier`. As a query strategy, we select the well-known uncertainty sampling (see line 12) as an example [Lewis and Gale, 1994]. It selects those instances the classifier is most uncertain about. Here, the uncertainty of an instance is measured by its class entropy.

**Algorithm 1 Active Learning Cycle**

```python
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import make_classification
from skactiveml.pool import UncertaintySampling
from skactiveml.utils import is_unlabeled, MISSING_LABEL

# Initializing the learner
X, y_true = make_classification(n_features=2, n_redundant=0)
y = np.full(shape=y_true.shape, fill_value=MISSING_LABEL)
clf = SklearnClassifier(LogisticRegression(),
                        classes=np.unique(y_true))
qs = UncertaintySampling(clf, method='entropy')

# Query labels and update classifier
n_cycles = 20
for c in range(n_cycles):
    unlbld_idx = np.where(is_unlabeled(y))[0]
    X_cand = X[unlbld_idx]
    query_idx = unlbld_idx[qs.query(X_cand=X_cand, X=X, y=y,
                                   batch_size=1)]
    y[query_idx] = y_true[query_idx]
    clf.fit(X, y)
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2021
doi:10.20944/preprints202103.0194.v1
We choose to acquire 20 labels ($n$ _cycles_ times _batch_size_) which means that we need to loop 20 times. The function _is_unlabeled_ in line 17 gives a Boolean mask indicating missing labels of $y$. The indices of the true values are stored in _unlbd_idx_, which are determined using the np._where_ function. Then, we create a list of labeling candidates (X_cand) in line 18, which includes the unlabeled instances of our data set. The number of data points to be selected from X_cand is controlled by the variable called _batch_size_, which is an argument of the query strategy in line 19. Next, we call the query function of our query strategy to receive the index of the best unlabeled data point (_batch_size_=1) from X_cand (see line 19). Note that we need to get the index with respect to $y$. Then, we overwrite the _MISSING_LABEL_ in $y$ with the true label of the selected instance. This can be seen as the label request from the oracle. Finally, we fit our classifier in line 22 with the new data. We continue to loop around until we reach the maximum number of iterations (_n_cycles_).

So far, we implemented the following methods (more will be added soon):

– Random Sampling
– Uncertainty Sampling Under Expected Precision [Wang et al., 2018]
– Epistemic Uncertainty Sampling [Nguyen et al., 2019]
– Active Learning with Cost Embedding [Huang and Lin, 2016]
– Expected Error Reduction [Roy and McCallum, 2001, Margineantu, 2005]
– Query by Committee [Seung et al., 1992]
– 4DS [Reitmaier and Sick, 2013]
– Multi-class Probabilistic Active Learning [Kottke et al., 2016]

### 4 Outlook

In the future, we are planning to incorporate more active learning scenarios and tools. Topics that we will be focusing on in the future are listed below.

– Stream-Based Active Learning [Zliobaitė et al., 2014]: Instances from a potentially non-stationary and unbounded data stream are processed one at a time. It will then be decided if the instance's label should be queried or discarded. Hence, performing a query for that instance will not be possible in the future. To restrict the number of queries, a budget is defined, limiting the number of queries in a given time frame. Such an active learning strategy aims to infer a well-performing model and maintain or improve the performance over time.

– Membership Query Synthesis [Settles, 2012]: Contrary to pool-based and stream-based Active Learning, membership query synthesis is not restricted to predefined data instances to query a label. The query strategy may query a label for any valid feature combination in a given feature space. Hence, such a query strategy aims to identify data points such that a well-performing model can be inferred with the least amount of queries.
Active Learning with Multiple Annotators [Zhang et al., 2015, Calma et al., 2016]: We use multiple annotators that label the data and assume that they are not omniscient but will label benevolently. The cost between each annotator may vary. Each query consists of a tuple with an annotator and the instance whose label is queried. The query strategy may query an instance's label multiple times to assess different annotators or verify past assessments.

Regression Tasks [Kumar and Gupta, 2020]: We also plan to include active learning strategies for optimizing the inference of regression models since the current framework only supports active learning for classification models.

Stopping Criteria [Vlachos, 2008]: We plan to incorporate stopping criteria for pool-based active learning. These often use heuristics to balance the number of queries against the predicted performance.

Evaluation and Visualization Tools/Methods [Kottke et al., 2017]: To better understand and compare the implemented active learning strategies, we will develop tools to simplify visualization and evaluation of Active Learning strategies.

5 Conclusion

The library scikit-activeml aims at bridging practitioners and researchers, unifying active learning processes, evaluation, and visualization, and should enable comparability across different approaches in the future. If you share this goal, please join our project group and be a part of the development team.

References


