SoftwareX 26 (2024) 101728

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

QuantificationLib: A Python library for quantification and prevalence estimation

Alberto Castaño, Jaime Alonso, Pablo González [*], Pablo Pérez, Juan José del Coz

Artificial Intelligence Center (University of Oviedo), Campus de Viesques, 33204, Gijón, Spain

ARTICLE INFO

Keywords:
Quantification learning
Prevalence estimation
Ordinal quantification

ABSTRACT

QuantificationLib is an open-source Python library that provides a comprehensive set of algorithms for quantification learning. Quantification, also known as prevalence estimation, is a supervised machine-learning task where the objective is to train a model that is able to predict the distribution of classes in a set of unseen examples or bags. This library offers a wide variety of quantification methods suited for easy prototyping and experimentation, applicable to a wide range of quantification applications.

Code metadata

Current code version	v0.1.2
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-24-00060
Permanent link to Reproducible Capsule	
Legal Code License	GNU General Public License v3.0
Code versioning system used	git
Software code languages, tools, and services used	python
Compilation requirements, operating environments & dependencies	scikit-learn, cvxpy, quadprog, scipy, numpy, ecos, statsmodels, pandas, matplotlib
If available Link to developer documentation/manual	https://aicgijon.github.io/quantificationlib/
Support email for questions	gonzalezgpablo@uniovi.es

1. Motivation and significance

Quantification learning is a relatively new task that has gained increasing attention in recent years due to its practical applications in various fields. Unlike classification, where the focus is on predicting the labels of unseen examples in a test set, the goal of quantification learning is to predict the aggregated results in the form of class label proportions in unseen test bags. It has many applications, for instance, monitoring the prevalence of plankton taxonomic groups in water samples, estimating the proportion of negative, positive and neutral reviews about certain products, predicting the prevalence of different types of customer calls in a customer service centre, etc. Quantification differs from classification, as the focus is not on the individual labels but on aggregated results [1]. While classification has been extensively studied and applied in various fields, quantification has emerged as a task in its own right, with distinct goals, methods, techniques, and evaluation measures. We refer the reader to the following Refs. [2,3] for seminal works on quantification learning.

In the field of quantification, various high-quality libraries are available (QuaPy [4], Qunfold [5] and QFy [6]). Notably, QuantificationLib distinguishes itself through several characteristics:

1. It emphasizes efficiency, particularly evident in experiments involving the comparison of multiple quantifiers using underlying classifiers. This efficiency is achieved by training the underlying classifiers only once and reusing it among several quantifiers.
2. It offers the capability to evaluate methods using synthetic data, for instance when working with data from known distributions, where the probability of belonging to a certain class is predetermined.
3. It utilizes optimization techniques to solve distribution matching-based quantification algorithms [7] across various distance metrics. When optimization is not feasible, an efficient search algorithm is employed.

[*] Corresponding author.
E-mail address: gonzalezgpablo@uniovi.es (Pablo González).

https://doi.org/10.1016/j.softx.2024.101728
Received 31 January 2024; Received in revised form 11 March 2024; Accepted 29 March 2024
Available online 8 April 2024

Table 1
Available quantification methods in QuantificationLib.

Category	Package	Methods
Binary	quantificationlib.binary	DeBias [9] SORD [10] QuantY [11]
Multiclass	quantificationlib.baselines	Classify & Count (CC) Adjusted Count (AC) [12,13] Probabilistic Classify & Count (PCC) [14] Probabilistic Adjusted Count (PAC) [14]
	quantificationlib.multiclass.df	HDx, HDy [15] Mixture Model (MM) [16] DFx, DFy [7,10]
	quantificationlib.multiclass.em	EM [17,18]
	quantificationlib.multiclass.energy	EDx [19] CvMy, EDy [20]
	quantificationlib.multiclass.friedman	FriedmanME [9]
	quantificationlib.multiclass.knn	PWKQuantifier [21]
Ordinal	quantificationlib.ordinal.ac	ACOrdinal [12]
	quantificationlib.ordinal.pdf	PDFOrdinaly [22]
	quantificationlib.ordinal.trees	OrdinalQuantificationTree [23]
Decomposition	quantificationlib.decomposition.multiclass	OneVsRestQuantifier
	quantificationlib.decomposition.ordinal	FrankAndHallQuantifier [24]
Ensembles	quantificationlib.ensembles	EoQ [25,26]

2. Software description

QuantificationLib is an open-source quantification library written in Python. It implements a pretty exhaustive list of quantification methods including binary, multiclass and ordinal methods, with support also for ensembles of quantification methods. It has been designed with ease of use in mind, keeping a lot of similarities in its structure with other machine learning libraries as *scikit-learn* [8]. It has a comprehensive and well-structured documentation,[1] that allows practitioners to quickly get started, with just a few lines of code.

2.1. Software architecture

QuantificationLib is structured as a Python library with a modular and extensible software architecture, providing a versatile framework for quantification learning tasks. The library is organized into distinct modules based on the type of quantification methods, encompassing binary, multiclass, ordinal, decomposition, and ensemble categories. Each module encapsulates specific algorithms, facilitating ease of navigation, maintenance, and the addition of new methods. The library adopts object-oriented principles to enhance modularity and reuse shared functionality among the methods of the same category. This approach facilitates a straightforward avenue for quantification practitioners to implement new quantification methods and seamlessly integrate them into the library.

In order to maintain high quality standards, a continuous integration (CI) system has been set up which ensures automatically that all tests are passed before making new releases. Also the documentation is built and deployed automatically to avoid any inconsistencies between code changes and documentation.

2.2. Software functionalities

QuantificationLib offers a broad range of quantification methods, which are systematically categorized using a simple and easy-to-understand taxonomy. These methods are divided into five distinct categories summarized in Table 1. The first category is the *binary* category, which includes methods that exclusively handle binary quantification problems. The second category is the *multiclass* category, consisting of methods that can handle quantification problems with any number of classes. The remaining categories include the *ordinal* category, which includes methods designed for ordinal quantification problems, the *decomposition* category, which includes decomposition methods for both multiclass and ordinal use cases, and the *ensembles* category, which provides a method for building ensembles of quantifiers. The library includes generalized versions of some methods. For instance, following the ideas in [7,10], DFx and DFx methods generalize HDx and HDy respectively allowing different representations for the distributions (PDFs or CDFs) and any loss functions (e.g. L1, L2, Topsøe) instead of just using the combination of PDFs and the Hellinger Distance. Interestingly, Mixture Model and SORD are instances of DFy using the combination of CDFs and L1, see [11]. The loss functions can be also optimized with AC, PACC and ACOrdinal when the confusion matrix computed in the training phase is not invertible.

3. Illustrative examples

3.1. Installation and basic model training pipeline

QuantificationLib is hosted on GitHub.[2] Packages for Python 3 are available through pip, using `pip install quantificationlib`.

To facilitate application by machine learning practitioners, the design of the library follows *scikit-learn* structure where each model provides a `fit` and a `predict` method. The standard workflow is: (1) create a new quantifier, instantiating the appropriate class; (2) train it using the `fit` method and, (3) predict unlabeled bags using the `predict` method, which will return a vector (with size equal to the number of classes) containing the class prevalence values for the bag passed as parameter.

[1] https://aicgijon.github.io/quantificationlib

[2] https://github.com/AICGijon/quantificationlib

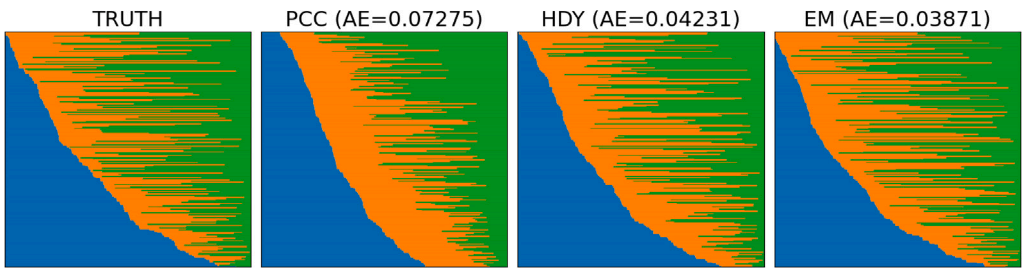

Fig. 1. Graphical experiment results for 200 test bags on the sample dataset. Each horizontal line corresponds to a single testing bag, each color represents a different class and the length of the color represents the proportion of that class. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

```
1  #base classifier
2  estimator = LogisticRegression()
3
4  #training Adjusted Count and EM quantifiers
5  ac = AC(estimator_train=estimator, estimator_test=estimator)
6  ac.fit(X_train, y_train)
7  em = EM(estimator_train=estimator, estimator_test=estimator)
8  em.fit(X_train, y_train)
9
10 #predicting a test sample
11 print(ac.predict(X=X_test))
12 print(em.predict(X=X_test))
```

Listing 1: Basic training/test quantification pipeline

Some quantification methods need the predictions of an underlying classifier in order to work. All these methods inherit from a common class `WithClassifiers` that unifies them under the same class hierarchy. Any classifier will work as long as it follows the *scikit-learn* standard, that is having the methods `fit`, `predict` and `predict_proba` (in case the probabilities are needed by the quantification method). On the other hand, methods inheriting from `WithoutClassifiers` class do not work with an underlying classifier so they do not require an estimator at all.

The library has been designed with efficiency in mind. This means that when a method uses an underlying classifier, the library will check if it is already trained or not. This saves the time of retraining multiple classifiers and provides a method to compare different quantifiers using the same base classifier. For instance, in Listing 1, *LogisticRegression* estimator is not trained again when EM quantifier is fitted in line 8. In addition, instead of working with estimators, the quantifiers under the `WithClassifiers` class can also work with raw probabilities, using the parameters `predictions_train` and `predictions_test` instead of `estimator_train` and `estimator_test`. These parameters are useful at least in two use cases: (1) to test quantifiers using controlled scenarios, where probabilities can be drawn, for instance, from some known probability distribution and (2) to make more flexible the training pipeline as probabilities can be computed beforehand and then passed to several quantification methods. Listing 2 presents an example of this latter case, where the predictions of all testing examples are computed only once (line 3) and then passed to two quantifiers (lines 5 and 7).

3.2. *Evaluating and comparing quantifiers*

The library makes it easy to compare and evaluate different quantifiers [27]. Under the package `quantificationlib.metrics` there is the implementation of the most used quantification error functions, for binary or multiclass problems, as *Absolute Error (AE)*, *Relative Absolute Error (RAE)*, *Mean Squared Error (MSE)* or *Kullback–Leibler Divergence (KLD)* among others. We also provide metrics for ordinal quantification problems such as the *Earth Moving Distance (EMD)*. Complementing error measures, the library also includes tools for artificially generating test bags which simplifies the evaluation of quantifiers. The following code shows a possible example of a typical evaluation pipeline:

```
1  bag_gen = PriorShift_BagGenerator(n_bags=10, bag_size=100)
2  prev_true, indexes = bag_gen.generate_bags(X_test, y_test)
3  preds = estimator.predict_proba(X_test)
4  for n_bag in range(10):
5      p_hat_ac = ac.predict(
6          X=None,
7          predictions_test=preds[indexes[:, n_bag], :],
8      )
9      mae = mean_absolute_error(prev_true[:, n_bag], p_hat_ac)
10     print("AC MAE=%f" % mae)
11     p_hat_em = em.predict(
12         X=None,
13         predictions_test=preds[indexes[:, n_bag], :],
14     )
15     mae = mean_absolute_error(prev_true[:, n_bag], p_hat_em)
16     print("EM MAE=%f" % mae)
```

Listing 2: Basic evaluation pipeline

Under the `quantificationlib.bag_generator` package, the user can find the implementation for different bag generators, that simulate different types of shift in the test bags, as *prior probability shift*, *covariate shift*, etc.

Finally, in the package `quantificationlib.plot` the user can find useful functions to plot the results and analyze them graphically. An example can be seen in Fig. 1.

4. Impact

Quantification is the field that works on training class prevalence estimators capable of predicting the prevalence of classes within an unlabeled set of examples. Applications are manifold and the field is growing rapidly, attracting researchers who seek to analyze aggregated results rather than individual labels. However, conducting experimentation for quantification papers poses challenges due to the non-trivial implementation of various quantification methods. Addressing this issue, QuantificationLib introduces an alternative experimental framework to others that exist in the field [4–6], incorporating a wide range of quantification methods, a fast and efficient implementation that also prioritizes ease of use. QuantificationLib extends its utility beyond research applications, finding practical use in diverse fields such as marine ecology [28], sentiment analysis [29], epidemiology [30], etc.

5. Conclusions

In this paper, we introduced QuantificationLib, a library for quantification learning which allows practitioners to build end-to-end quantification pipelines with just a few lines of Python code. The library is published under an Open Source licence which makes a significant contribution to the quantification community allowing researchers to easily test and compare different quantification methods. Due to its design and structure, the library easily accommodates the integration of new methods that may arise in the field of quantification.

CRediT authorship contribution statement

Alberto Castaño: Writing – review & editing, Software. **Jaime Alonso:** Software, Visualization. **Pablo González:** Project administration, Software, Writing – original draft. **Pablo Pérez:** Software, Writing – review & editing. **Juan José del Coz:** Software, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Pablo Gonzalez reports article publishing charges, equipment, drugs, or supplies, travel, and writing assistance were provided by University of Oviedo. Alberto Castano reports article publishing charges, equipment, drugs, or supplies, travel, and writing assistance were provided by University of Oviedo. Jaime Alonso reports article publishing charges, equipment, drugs, or supplies, travel, and writing assistance were provided by University of Oviedo. Pablo Perez reports article publishing charges, equipment, drugs, or supplies, travel, and writing assistance were provided by University of Oviedo. Juan Jose del Coz reports article publishing charges, equipment, drugs, or supplies, travel, and writing assistance were provided by University of Oviedo. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

All the data used is public and is available for download.

Acknowledgments

This work was supported by grant PID2019-110742RB-I00 from Spanish Ministerio de Economía y Competitividad (MINECO) and grant PID2019-109238GB-C21 from Spanish Ministry of Science and Innovation.

References

[1] González P, Díez J, Chawla N, del Coz JJ. Why is quantification an interesting learning problem? Progr Artif Intell 2017;6:53–8.
[2] González P, Castaño A, Chawla NV, Coz JJD. A review on quantification learning. ACM Comput Surv 2017;50(5):1–40.
[3] Esuli A, Fabris A, Moreo A, Sebastiani F. Learning to quantify. In: The information retrieval series, vol. 47, Springer; 2023, http://dx.doi.org/10.1007/978-3-031-20467-8.
[4] Moreo A, Esuli A, Sebastiani F. QuaPy: A Python-based framework for quantification. In: Proceedings of the 30th ACM international conference on information & knowledge management. 2021, p. 4534–43.
[5] Bunse M. qunfold: Composable quantification and unfolding methods in Python. In: Proceedings of the 3rd international workshop on learning to quantify (LQ 2023), co-located at ECML-pKDD. 2023, p. 1–7.
[6] Schumacher T, Strohmaier M, Lemmerich F. A comparative evaluation of quantification methods. 2021, arXiv preprint arXiv:2103.03223.
[7] Firat A. Unified framework for quantification. 2016, arXiv preprint arXiv:1606.00868.
[8] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in Python. J Mach Learn Res 2011;12:2825–30.
[9] Friedman J. Class counts in future unlabeled samples (detecting and dealing with concept drift). 2014, Presentation at MIT CSAIL Big Data Event.
[10] Maletzke A, dos Reis D, Cherman E, Batista G. Dys: a framework for mixture models in quantification. In: Proceedings of the AAAI conference on artificial intelligence. Vol. 33, 2019, p. 4552–60.
[11] Castaño A, Alonso J, González P, del Coz JJ. An equivalence analysis of binary quantification methods. In: Proceedings of the AAAI Conference on Artificial Intelligence. 37, (6):2023, p. 6944–52.
[12] Forman G. Quantifying counts and costs via classification. Data Min Knowl Discov 2008;17:164–206.
[13] Lipton Z, Wang Y-X, Smola A. Detecting and correcting for label shift with black box predictors. In: International conference on machine learning. PMLR; 2018, p. 3122–30.
[14] Bella A, Ferri C, Hernández-Orallo J, Ramirez-Quintana MJ. Quantification via probability estimators. In: 2010 IEEE international conference on data mining. IEEE; 2010, p. 737–42.
[15] González-Castro V, Alaiz-Rodríguez R, Alegre E. Class distribution estimation based on the Hellinger distance. Inform Sci 2013;218:146–64.
[16] Forman G. Counting positives accurately despite inaccurate classification. In: Gama J, Camacho R, Brazdil PB, Jorge AM, Torgo L, editors. Machine learning: ECML 2005. Berlin, Heidelberg: Springer; 2005, p. 564–75.
[17] Saerens M, Latinne P, Decaestecker C. Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure. Neural Comput 2002;14(1):21–41.
[18] Alexandari A, Kundaje A, Shrikumar A. Maximum likelihood with bias-corrected calibration is hard-to-beat at label shift adaptation. In: International conference on machine learning. PMLR; 2020, p. 222–32.
[19] Kawakubo H, Du Plessis MC, Sugiyama M. Computationally efficient class-prior estimation under class balance change using energy distance. IEICE Trans Inf Syst 2016;99(1):176–86.
[20] Castaño A, Morán-Fernández L, Alonso J, Bolón-Canedo V, Alonso-Betanzos A, del Coz J. A theoretical analysis of quantification methods based on matching distributions. University of Oviedo; 2021, https://github.com/bertocast/adjust_dist_xy.
[21] Barranquero J, González P, Díez J, Del Coz JJ. On the study of nearest neighbor algorithms for prevalence estimation in binary problems. Pattern Recognit 2013;46(2):472–82.
[22] Castaño A, González P, González JA, Del Coz JJ. Matching distributions algorithms based on the earth mover's distance for ordinal quantification. IEEE Trans Neural Netw Learn Syst 2022.
[23] Da San Martino G, Gao W, Sebastiani F. Ordinal text quantification. In: Proceedings of the 39th international ACM SIGIR conference on research and development in information retrieval. 2016, p. 937–40.
[24] Frank E, Hall M. A simple approach to ordinal classification. In: Machine learning: ECML 2001: 12th European conference on machine learning Freiburg, Germany, September 5–7, 2001 proceedings. Vol. 12. Springer; 2001, p. 145–56.
[25] Pérez-Gállego P, Quevedo JR, del Coz JJ. Using ensembles for problems with characterizable changes in data distribution: A case study on quantification. Inf Fusion 2017;34:87–100.
[26] Pérez-Gállego P, Castano A, Quevedo JR, del Coz JJ. Dynamic ensemble selection for quantification tasks. Inf Fusion 2019;45:1–15.
[27] Sebastiani F. Evaluation measures for quantification: An axiomatic approach. Inf Retrieval J 2020;23(3):255–88.
[28] González P, Castaño A, Peacock EE, Díez J, Del Coz JJ, Sosik HM. Automatic plankton quantification using deep features. J Plankton Res 2019;41(4):449–63.
[29] Esuli A, Moreo Fernández A, Sebastiani F. A recurrent neural network for sentiment quantification. In: Proceedings of the 27th ACM international conference on information and knowledge management. 2018, p. 1775–8.
[30] Baccianella S, Esuli A, Sebastiani F. Variable-constraint classification and quantification of radiology reports under the ACR index. Expert Syst Appl 2013;40(9):3441–9.