On Reproducible Implementations in Unsupervised Concept Drift Detection Algorithms Research

Daniel Lukats[1,2](✉) and Frederic Stahl[1]

[1] German Research Center for Artificial Intelligence, Marie-Curie-Straße 1, 26129 Oldenburg, Germany
{Daniel.Lukats,Frederic_Theodor.Stahl}@dfki.de
[2] Carl-von-Ossietzky-Universität Oldenburg, Ammerländer Heerstraße 114-118, 26129 Oldenburg, Germany

Abstract. In order to create reproducible experimentation and algorithms in machine learning and data mining research, reproducible descriptions of the algorithms are needed. These can be in the form of source code, pseudo code and prose. Efforts in academia commonly focus on accessibility of source code. Based on an internal study reproducing unsupervised concept drift detectors, this work argues that a publication's content is equally important and highlights common issues affecting attempts at implementing unsupervised concept drift detectors. These include major issues prohibiting implementation entirely, as well as minor issues, which demand increased effort from the developer. The paper proposes the use of a checklist as a consistent tool to ensure better quality and reproducible publications of algorithms. The issues highlighted in this work could mark a starting point, although future work is required to ensure representation of more diverse areas of research in artificial intelligence.

Keywords: Reproducibility · Pseudo code · Data mining

1 Introduction

Reproducible research is a core tenet of the scientific method. In contrast with natural sciences, computer science offers a unique benefit in this regard, as source code and data sets can be published on platforms such as GitHub or GitLab to provide access to the experiment. Accordingly, there is a lot of effort in academia to encourage researchers to publish their code alongside their papers and to ensure certain quality standards [8,11,14]. Furthermore, publishers and conference organisers address reproducibility more generally, addressing issues affecting reproducibility such as pseudo-randomness [11] or discussing the topic of reproducibility in machine learning itself [9,12]. Various publications deal with reproducibility in their respective domain of machine learning, e.g. [3,7].

However, there is little discussion about pseudo code in the literature, with the exception of the research area of pseudo code generation and an analysis about the psychological effects of pseudo code [1]. Most online resources are similarly one-sided, explaining possible syntax for pseudo code in brief tutorials—various blog posts and documents provided by university lecturers are available online.

This work argues that publishing source code alone is not sufficient for the reproducibility of machine learning and data mining algorithms. In various cases, access to good quality pseudo code might be more beneficial even when source code is available, as one might need to re-implement the algorithm. The reasons therefor may include:

- There are contradictions in the source code and the algorithm's description.
- The source code is not available in the desired programming language.
- The poor quality of the source code.
- The implementation is no longer maintained, causing errors due to version mismatches or use of deprecated functions for example.
- There is no license or the license does not match the intended use-case.

This paper is based on experiences made during an internal scientific study on unsupervised concept drift detectors for analysis of data streams, for which 23 publications were considered for implementation. In the following, the issues encountered are described—in both source code and algorithm descriptions—and how they affected the implementations. Finally, possible solutions to these issues are discussed.

2 Observed Issues

The issues detailed in the following and the analysis of the 23 publications were observed and conducted by the authors of this paper. It is acknowledged here that there may be bias in this work, since of some of the perceived issues may be subject to the background of the authors. Yet, these issues are present in publications related to artificial intelligence in general to varying degrees from the experience of the authors, although the extent of reproducibility issues may differ between various areas of research.

2.1 Issues with Source Code

In the aforementioned case study it was observed that only 8 papers out of 23 provided source code. First, the source of the 8 publications was examined. Although not all of the observed issues make reproduction impossible, they negatively affect the usability and may diminish the publication's impact preventing researchers reproducing its result. The occurrences of the following issues are summarised in Table 1:

1. Only three repositories contained a license detailing the terms of use. If no license is included, the work cannot be used by others as default copyright laws apply [5,6].

2. All but one repository contained incomplete instructions for installation or no information on required dependencies at all, thus making this step of reproducing the authors' experiments error-prone.
3. Merely two repositories contained documented source code, again increasing the risk of erroneous use of the provided repository.
4. Five out of eight implementations did not match the algorithm description in the corresponding publication. In most instances the implementation included normalisation steps, which were not mentioned in the paper. This can prove problematic in two different scenarios. On the one hand, if one attempts to reproduce a publication's result by implementing the algorithm ourselves, one may leave out a crucial component of the algorithm and therefore may not be able to achieve the desired results. On the other hand, if one decides to include the algorithm as a third-party library, one will not know that our data may be pre-processed by the algorithm already, e.g. data may accidentally be normalised twice, which can introduce errors.
5. Three publications provided incomplete access to the data sets used for evaluation. Hence, the results of these publications cannot be reproduced.
6. Seeds were given in 2 out of 4 publications, which used algorithms depending on pseudo-randomness. Without seeds reproducing exact results is impossible, although similar results should be achievable.

Table 1. 8 out of 23 publications published source code. The publications are number-coded. A ✗ indicates an issue in the respective column, e.g. a missing license or declarations of dependencies. A $\sim$ indicates that information is given but incomplete, as install instructions may be missing. Not all detectors include pseudo-random components; seeds were only considered for those which do.

Publication	License	Dependencies	Documentation	Consistency	Data streams	Seed
7	✗	✗	✗	✗	✓	—
8	✗	✗	✗	✗	✓	✗
12	✓	✗	✗	✓	$\sim$	✗
15	✗	$\sim$	✗	✗	✗	—
16	✓	✓	✓	✓	✗	✓
19	✗	✗	✗	✗	✓	✓
20	✓	✓	✓	✓	✓	—
22	✗	$\sim$	✗	✗	✓	—
% ✗	63%	50%	75%	63%	25%	50%

If one would like to use an algorithm that was published without a license, it must be implemented from scratch for legal reasons [5,6]. The other issues need to be weighed by the developers and users of said algorithm. Missing dependencies and a poor documentation may be on a subjective level irritating. However, if the code quality is sufficient otherwise, using the published repository may be reasonable. Similarly, undocumented implementation choices, seeds and incomplete data streams may not matter depending on the intended use case.

2.2 Issues with Publication Texts

When one decides to implement a published AI or data mining algorithm, from a scientific perspective the publication must provide all information required. Various issues can affect attempts, as missing information might make an authentic implementation impossible whereas other issues merely make these attempts more demanding. In the aforementioned internal study, the following issues were observed (see Table 2 for occurrences):

1. Only 15 publications provided pseudo code providing a higher level view of the algorithm's design. The remaining 8 papers are described in prose only, which can cause the implementation process to be cumbersome and error-prone, as the interplay of different components of the algorithm may not be obvious without a singular abstract overview. None of these 8 provided source code either.
2. Most crucially, 11 publications are incomplete and miss details. Some include undefined symbols and others lack information on some components of the algorithm, e.g. how to update data windows. In either case a faithful implementation of the algorithm is impossible, as one simply cannot know the intended behavior of undefined components.
3. Many concept drift detectors model data by estimating probability distributions. In 6 publications the required approximations are foregone; the publications describe only the ideal theoretical distributions. Often properties of probability distributions can be approximated with multiple methods, e.g. when estimating a variance [15]. Likewise, models such as Gaussian mixture models often feature different initialisation techniques [13]. Consequently, when implementing an algorithm one may need to make assumptions about the chosen estimators, thus introducing at least slight deviations from the original publication.
4. 3 publications provide no intra-document references in the form of equation numbers or otherwise. Admittedly this is a minor issue, although it demands further effort during implementation, since the correct definition needs to be searched for.
5. 7 publications included definitions in prose without any noteworthy highlight through typesetting or formatting like in-line code listings or equations formatted as in the LaTeX math mode. This is another minor issue that demands more effort by the developer implementing the algorithm.
6. Finally, in 4 publications the pseudo code and the publication text contradict each other. Out of experience, this ambiguity can only be resolved by testing all options.

Table 2. Different issues affected attempts at implementing 23 unsupervised concept drift detectors. The publications are number-coded. A ✗ indicates an issue in the respective column. Empty cells indicate no issue.

Publication	Pseudo Code	Complete Definitions	Approximations	References	Prose Definitions	Consistency
1	✗					
2		✗				
3		✗	✗	✗	✗	✗
4	✗					
5	✗	✗		✗		
6		✗			✗	✗
7		✗				
8		✗				
9	✗		✗		✗	
10	✗				✗	
11	✗					
12					✗	
13		✗	✗			
14	✗					
15		✗				✗
16			✗			
17		✗				
18	✗		✗		✗	
19		✗				
20						
21		✗				
22			✗			
23				✗	✗	✗
% ✗	35%	48%	26%	13%	30%	17%

3 Discussion

The issues discussed in Sect. 2 were spotted neither by the authors nor by reviewers of the respective papers. A checklist would be the easiest solution; it can be consulted by both authors and reviewers. The issues highlighted in Sect. 2.2 could mark a starting point, although they are limited in two regards: Firstly, the perception of some of these issues is subjective, so more diverse perspectives are desired. Secondly, these issues are based on a small sample of 23 unsupervised concept drift detectors only. Though many of these issues will apply to machine learning and data mining publications in general, the list may be missing important issues from other areas of research. Since pseudo code is not formalized like programming languages, tools like linters would be difficult to establish. Other solutions to improve the understanding of algorithms involve literate programming [4], or more recently literate computing [10]. But these solutions depend on the availability of source code. Checklists can be powerful tools improving efficiency and consistency [2]. They are easier to develop and implement than complex tools, a checklist might be a suitable start to discuss reproducible pseudo code. However, more work is required to ensure diverse backgrounds and areas of research in artificial intelligence are represented. Therefore, experiences from other research groups in the areas of machine learning and data mining will be incorporated into a more comprehensive study in the future.

Acknowledgements. This paper has received partial funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101000825 (NAUTILOS).This work also received partial funding from Niedersächsisches Vorab under grant number *ZN3683* (ChESS).

References

1. Bellamy, R.K.: What does pseudo-code do? A psychological analysis of the use of pseudo-code by experienced programmers. Hum. Comput. Interact. **9**(2), 225–246 (1994)
2. Gawande, A.: The Checklist Manifesto. Metropolitan Books, New Delhi (2009)
3. Heil, B.J., Hoffman, M.M., Markowetz, F., Lee, S.I., Greene, C.S., Hicks, S.C.: Reproducibility standards for machine learning in the life sciences. Nat. Meth. **18**(10), 1132–1135 (2021). https://doi.org/10.1038/s41592-021-01256-7
4. Knuth, D.E.: Literate programming. Comput. J. **27**(2), 97–111 (1984). https://doi.org/10.1093/comjnl/27.2.97
5. Licensing a Repository. https://docs.github.com/en/repositories/managing-your-repositorys-settings-and-features/customizing-your-repository/licensing-a-repository. Accessed 26 June 2023
6. No License | Choose a License. https://choosealicense.com/no-permission. Accessed 26 June 2023
7. Olorisade, B.K., Brereton, P., Andras, P.: Reproducibility of studies on text mining for citation screening in systematic reviews: evaluation and checklist. J. Biomed. Inform. **73**, 1–13 (2017). https://doi.org/10.1016/j.jbi.2017.07.010
8. Papers with Code. https://paperswithcode.com. Accessed 26 June 2023
9. Pineau, J., et al.: Improving reproducibility in machine learning research (a report from the NeurIPS 2019 reproducibility program). J. Mach. Learn. Res. **22**, 7459–7478 (2021)
10. Pérez, Fernando, K.J.M.: Developing open-source scientific practice. In: Implementing Reproducible Research. Chapman and Hall/CRC (2014)
11. Reproducibility Checklist | AAAI 2023 Conference. https://aaai-23.aaai.org/reproducibility-checklist. Accessed 26 June 2023
12. Reproducibility in Machine Learning. https://sites.google.com/view/icml-reproducibility-workshop/home. Accessed 26 June 2023
13. Shireman, E., Steinley, D., Brusco, M.J.: Examining the effect of initialization strategies on the performance of Gaussian mixture modeling. Behav. Res. Meth. **49**(1), 282–293 (2016). https://doi.org/10.3758/s13428-015-0697-6
14. Tips for Publishing Research Code. https://github.com/paperswithcode/releasing-research-code. Accessed 26 June 2023
15. Wolter, K.M.: Introduction to Variance Estimation. Statistics for Social and Behavioral Sciences, Springer, New York, NY (2007). https://doi.org/10.1007/978-0-387-35099-8