The Journal of Open Source Software

ReLax: Efficient and Scalable Recourse Explanation Benchmark using JAX

Hangzhi Guo[1¶], **Xinchang Xiong**[2], **Wenbo Zhang**[1], and **Amulya Yadav**[1]

1 Penn State University, USA **2** Duke University, USA ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

Counterfactual explanation[1] techniques provide contrast cases to individuals adversely affected by ML preictions. For instance, recourse methods can provide suggestions for loan applicants who have been rejected by a bank's ML algorithm, or give practical advice to teachers handling students at risk of dropping out. Numerous recourse explanation methods have been recently proposed. Yet, current research practice focuses on medium-sized datasets (typically around ~50k data points). This limitation impedes the progress in algorithmic recourse and raises concerns about the scalability of existing approaches.

To address this challenge, we propose `ReLax`, a JAX-based benchmarking library, designed for efficient and scalable recourse explanations. `ReLax` supports various recourse methods and datasets, demonstrating performance improvements of at least two orders of magnitude over current libraries. Notably, ReLax can benchmark real-world datasets up to 10 million data points, a 200-fold increase over existing norms, without imposing prohibitive computational costs.

Statement of need

Recourse and counterfactual explanation methods concentrate on the generation of new instances that lead to contrastive predicted outcomes (Karimi et al., 2020; Stepin et al., 2021; Verma et al., 2020). Given their ability to provide actionable recourse, these explanations are often favored by human end-users (Bhatt et al., 2020; Binns et al., 2018; Miller, 2019).

Despite progress made in counterfactual explanation research (Guo, Jia, et al., 2023; Guo, Nguyen, et al., 2023; Mothilal et al., 2020; Upadhyay et al., 2021; Ustun et al., 2019; Vo et al., 2023; Wachter et al., 2017), current research practices often restrict the evaluation of recourse explanation methods on medium-sized datasets (with under 50k data points). This constraint primarily stems from the excessive runtime overhead of recourse generation by the existing open-source recourse libraries (Klaise et al., 2021; Mothilal et al., 2020; Pawelczyk et al., 2021). For instance, as shown in Figure 1, the CARLA library (Pawelczyk et al., 2021) requires roughly 30 minutes to benchmark the adult dataset containing $\sim 32,000$ data points. At this speed, it would take CARLA approximately 15 hours to benchmark a dataset with 1 million samples, and nearly one week to benchmark a 10-million dataset. Consequently, this severe runtime overhead hinders the large-scale analysis of recourse explanations and the research development of new recourse methods.

[1] Counterfactual explanation (Wachter et al., 2017) and algorithmic recourse (Ustun et al., 2019) share close connections (Stepin et al., 2021; Verma et al., 2020), which leads us to use these terms interchangeably

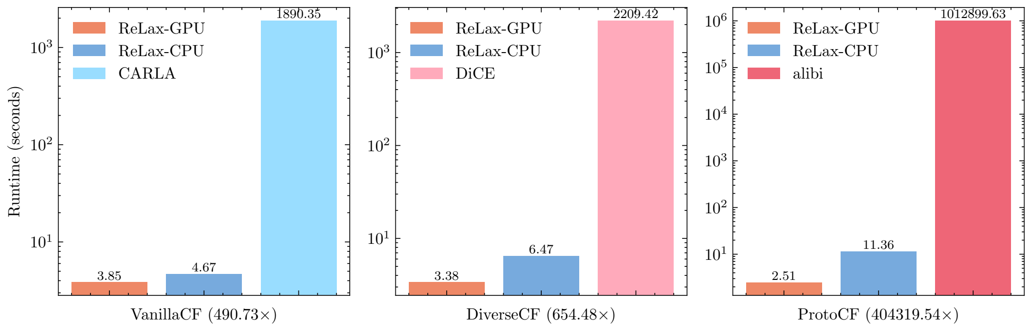


Figure 1: Runtime comparison of the *adult* dataset between `ReLax` and three open-source recourse librarires (CARLA (Pawelczyk et al., 2021), DiCE (Mothilal et al., 2020), and alibi (Klaise et al., 2021).

In this work, we present `ReLax` (**Re**course Explanation **L**ibrary using J**ax**), the *first* recourse explanation library in JAX (Bradbury et al., 2018; Frostig et al., 2018). Our contributions are three-fold:

- (Fast and Scalable System) `ReLax` is an *efficient and scalable benchmarking library* for recourse and counterfactual explanations.
- (Comprehensive set of Methods) `ReLax` implements 9 recourse explanation methods. In addition, `ReLax` include 14 medium-sized datasets, and one large-scale dataset.
- (Extensive Experiments) We perform comprehensive experiments on both medium-sized and large-sized datasets, which showcases the usability and scalability of the library.

Efficiency and Scalability in `ReLax`

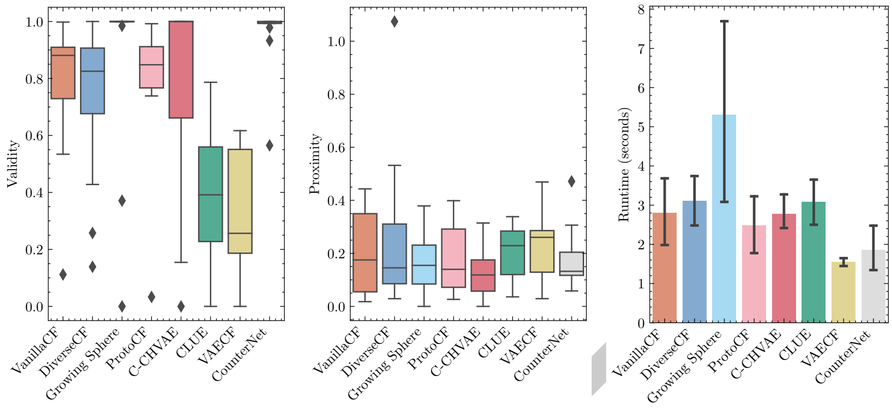
`ReLax` supports three recourse generation strategies: *sequential*, *vectorized*, and *parallelized* strategy. In particular, the *sequential* generation strategy is inefficient, albeit being adopted in most existing libraries. On the other hand, the *vectorized* and *parallelized* strategies play a vital role in equipping `ReLax` to benchmark large-scale datasets with a practical computational cost. In addition to these, `ReLax` further enhances its performance by fusing inner recourse generation steps via the Just-In-Time (JIT) compilation. Together, `ReLax` ensures efficient and scalable performance across diverse data scales and complexities.

Recourse Methods & Datasets

`ReLax` implements nine recourse methods using JAX including (i) three non-parametric methods (VanillaCF (Wachter et al., 2017), DiverseCF (Mothilal et al., 2020), GrowingSphere (Laugel et al., 2017)); (ii) three semi-parametric methods (ProtoCF (Van Looveren & Klaise, 2019), C-CHVAE (Pawelczyk et al., 2020), CLUE (Antoran et al., 2021)); and (iii) three parametric methods (VAE-CF (Mahajan et al., 2019), CounterNet (Guo, Nguyen, et al., 2023), L2C (Vo et al., 2023)).

Furthermore, we gather 14 medium-sized binary-classification tabular datasets. We also benchmark over the forktable dataset (Ding et al., 2021) for predicting individuals' annual income. This US censuring dataset contains ~ 10 million data points. To our knowledge, this is the first attempt to benchmark a dataset at the scale of 10 million data points in the recourse explanation community.

The Journal of Open Source Software



(a) Boxplot of validity on medium-size datasets for each recourse method. High validity is desirable.

(b) Boxplot of normalized proximity on medium-sized datasets. Low proximity is preferable.

(c) Barplot of runtime on medium-size datasets for each recourse method. Low runtime is desirable.

Figure 2: Comparison of recourse method performance across 14 medium-sized datasets. It is desirable to achieve *high* validity, *low* proximity, and *low* runtime.

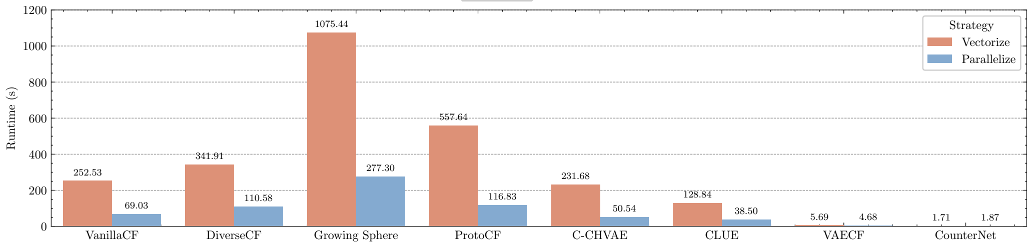


Figure 3: Runtime comparison of different recourse generation strategies on the forktable dataset.

Experimental Results

Figure 2 compares the validity, proximity, and runtime achieved by nine recourse methods averaged on 14 medium-sized datasets. In particular, validity and proximity measure the quality of the generated counterfactual explanations. We observe that CounterNet and Growing Sphere achieve the best validity score, and C-CHVAE achieves the best proximity score. In terms of runtime, all recourse methods complete the entire recourse generation process within 10 seconds, while CounterNet and VAECF outperform others by completing under 2 seconds.

Figure 3 compares the runtime for each recourse explanation method in adopting the vectorized and parallelized strategies on the forktable dataset (with 10M data points). First, ReLax is highly efficient in benchmarking the large-scale dataset, with the maximum runtime being under 30 minutes. On the other hand, by estimation, existing libraries should take at least one week to complete recourse generation on datasets at this scale. In addition, the parallelized strategy cuts the runtime by roughly 4X, which demonstrates that ReLax's potential in benchmarking even larger datasets.

References

Antoran, J., Bhatt, U., Adel, T., Weller, A., & Hernández-Lobato, J. M. (2021). Getting a {CLUE}: A method for explaining uncertainty estimates. *International Conference on Learning Representations*. https://openreview.net/forum?id=XSLF1XFq5h

Bhatt, U., Xiang, A., Sharma, S., Weller, A., Taly, A., Jia, Y., Ghosh, J., Puri, R., Moura, J. M. F., & Eckersley, P. (2020). Explainable machine learning in deployment. *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 648–657. https://doi.org/10.1145/3351095.3375624

Binns, R., Van Kleek, M., Veale, M., Lyngs, U., Zhao, J., & Shadbolt, N. (2018). 'It's reducing a human being to a percentage' perceptions of justice in algorithmic decisions. *Proceedings of the 2018 Chi Conference on Human Factors in Computing Systems*, 1–14.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018). *JAX: Composable transformations of Python+NumPy programs* (Version 0.4.10). http://github.com/google/jax

Ding, F., Hardt, M., Miller, J., & Schmidt, L. (2021). Retiring adult: New datasets for fair machine learning. *Advances in Neural Information Processing Systems*, *34*.

Frostig, R., Johnson, M. J., & Leary, C. (2018). Compiling machine learning programs via high-level tracing. *Systems for Machine Learning*, *4*(9).

Guo, H., Jia, F., Chen, J., Squicciarini, A., & Yadav, A. (2023). RoCourseNet: Robust training of a prediction aware recourse model. *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 619–628.

Guo, H., Nguyen, T., & Yadav, A. (2023). CounterNet: End-to-end training of prediction aware counterfactual explanation. *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA*. https://doi.org/10.1145/3580305.3599290

Karimi, A.-H., Barthe, G., Schölkopf, B., & Valera, I. (2020). A survey of algorithmic recourse: Definitions, formulations, solutions, and prospects. *arXiv Preprint arXiv:2010.04050*.

Klaise, J., Looveren, A. V., Vacanti, G., & Coca, A. (2021). Alibi explain: Algorithms for explaining machine learning models. *Journal of Machine Learning Research*, *22*(181), 1–7. http://jmlr.org/papers/v22/21-0017.html

Laugel, T., Lesot, M.-J., Marsala, C., Renard, X., & Detyniecki, M. (2017). Inverse classification for comparison-based interpretability in machine learning. *arXiv Preprint arXiv:1712.08443*.

Mahajan, D., Tan, C., & Sharma, A. (2019). Preserving causal constraints in counterfactual explanations for machine learning classifiers. *arXiv Preprint arXiv:1912.03277*.

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, *267*, 1–38.

Mothilal, R. K., Sharma, A., & Tan, C. (2020). Explaining machine learning classifiers through diverse counterfactual explanations. *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 607–617.

Pawelczyk, M., Bielawski, S., Heuvel, J. van den, Richter, T., & Kasneci, G. (2021). CARLA: A python library to benchmark algorithmic recourse and counterfactual explanation algorithms. *Advances in Neural Information Processing Systems Track on Datasets and Benchmarks*.

Pawelczyk, M., Broelemann, K., & Kasneci, G. (2020). Learning model-agnostic counterfactual explanations for tabular data. *Proceedings of the Web Conference 2020*, 3126–3132.

Stepin, I., Alonso, J. M., Catala, A., & Pereira-Fariña, M. (2021). A survey of contrastive and counterfactual explanation generation methods for explainable artificial intelligence. *IEEE Access*, *9*, 11974–12001.

Upadhyay, S., Joshi, S., & Lakkaraju, H. (2021). Towards robust and reliable algorithmic recourse. *arXiv Preprint arXiv:2102.13620*.

Ustun, B., Spangher, A., & Liu, Y. (2019). Actionable recourse in linear classification. *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 10–19.

Van Looveren, A., & Klaise, J. (2019). Interpretable counterfactual explanations guided by prototypes. *arXiv Preprint arXiv:1907.02584*.

Verma, S., Dickerson, J., & Hines, K. (2020). Counterfactual explanations for machine learning: A review. *arXiv Preprint arXiv:2010.10596*.

Vo, V., Le, T., Nguyen, V., Zhao, H., Bonilla, E. V., Haffari, G., & Phung, D. (2023). Feature-based learning for diverse and privacy-preserving counterfactual explanations. *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2211–2222.

Wachter, S., Mittelstadt, B., & Russell, C. (2017). Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.*, *31*, 841.