Highlighting hurdles to benchmarking machine learning methods

Célestin Eve[1,2], Thomas Moreau[1], Gaël Varoquaux[2]
[1]INRIA MIND
[2]INRIA SODA

Abstract

Benchmarking learning algorithms is a critical step in advancing machine learning research, providing a means to evaluate the effectiveness of new methods against established benchmarks. However, the benchmarking process is fraught with variability, which can lead to misleading conclusions if not carefully managed. In this study, we investigate the impact of various factors on benchmarking outcomes, including data splitting methods, noise levels, and dataset sizes. Through experiments using simulated data and the MNIST dataset, we demonstrate that these factors can significantly alter the performance rankings of algorithms. Our findings highlight the need for rigorous benchmarking practices to ensure reliable and robust comparisons in machine learning research.

Keywords: Benchmarking, Machine Learning, Ranking

1 Motivation/Introduction

Benchmarking learning algorithms is crucial for scientific progress in machine learning. Comparing a newly developed algorithm against state-of-the-art methods provides an early indication of the algorithm's effectiveness. To facilitate such comparisons, reference benchmarks have been established, such as ImageNet for image classification [1] or Atari 2600 games for reinforcement learning [2]. Additionally, competitive platforms like Kaggle have emerged, allowing for the straightforward comparison of algorithms on the same dataset, while using private leaderboards to mitigate the risk of overfitting. Tools like Benchopt [3] also enable efficient benchmarking of optimization methods.

Unfortunately, benchmarking is inherently a process that exhibits variability in numerous ways [4], which can result in inconsistent rankings and misleading conclusions. In the field of anomaly detection in time series, certain widely adopted benchmarking practices may even perform worse than random [5]. Therefore, it is essential to accurately identify and account for all sources potentially invalidating numerical comparisons in order to achieve reliable and robust results.

2 Methods

Let $S_{\mathcal{D}}$ denote the score function associated with a dataset $\mathcal{D}$, which takes as input a decision function g and returns its score on the dataset $\mathcal{D}$. The oracle score, S^*, is defined as the expected score over the population distribution d, i.e., $S^*(g) = \mathbb{E}_{\mathcal{D}\sim d}[S_{\mathcal{D}}(g)]$, where d represents the underlying population distribution.

Our approach involved two main steps: first, we compared decision functions, and then we benchmarked learning algorithms. We considered eleven

score estimators using different methods for splitting the data into training (the samples seen by the learning algorithm to deduce a decision fonction) and testing (the samples on which the performance of the learned decision function is tested) sets. The methods evaluated in our study include *train_test_split*, *KFold* and *ShuffleSplit* with a varying number of splits ranging from 2 to 10.

To assess the ranking stability of these different procedures, we first generated a simulated dataset with normally distributed features to calculate the optimal theoretical estimator, the Bayes estimator, with a score s_{Bayes} on the test set. By adding normally distributed noise, we created a suboptimal version with performance inversely related to the noise level. Noise in data generation also reduced the Bayes estimator's performance. The goal of benchmarking is to estimate the true ranking of methods, known as the oracle ranking. This setup allowed us to empirically estimate on the test set $\mathbb{P}\left(S_{\mathcal{I}_{test}}(\text{Bayes}) < S_{\mathcal{I}_{test}}(\text{noisy})\right)$, leading to a ranking inversion as $S^*(\text{Bayes}) > S^*(\text{noisy})$, depending on noise levels and dataset size.

Subsequently, we ran experiments on the MNIST dataset, introduced by [6]. We selected a subset of the MNIST training set, which consists of 60,000 samples, and referred to this subset as the study set, varying its size for different experiments. The study set was then divided into training, testing, and validation subsets. We compared the performance of decision functions generated by various learning algorithms (ExtraTrees, RandomForest, XGBoost) on the whole original MNIST training set, referred to here as the reference set.

Although the initial set of experiments does not include a learning phase, we use the same data splitting techniques to maintain consistency within our experiments. All experiments were implemented in Python, utilizing learning algorithms and data splitting procedures from the scikit-learn library [7], as well as the XGBClassifier from the XGBoost library [8].

3 Results

We ran each configuration of our learning-free experiments 200 times. In many cases, as can be seen in Table 1, the noisy estimator ranked as the best estimator more than 15% of the time.

On MNIST, in most experiments ran, the computed ranking of the learning algorithms varied depending on the size of the study set, all others things being equal. Typically, when the study set comprises more than 50% of the reference set, $S_{\mathcal{I}_{reference}}(\text{XGBoost}) > S_{\mathcal{I}_{reference}}(\text{ExtraTrees})$, but this ranking is reversed when the study set is reduced to 25% of the reference set.

4 Conclusion

Our experiments underscore the complexity and challenges inherent in benchmarking machine learning algorithms. The results reveal that factors such as the method of data splitting, the introduction of noise, and the size of the study set can significantly influence the performance rankings of different algorithms. Notably, even subtle variations in experimental configurations, such as the noise level or dataset size, can lead to substantial differences in outcomes, including

Procedure	Number of Splits	Number of Inversions	$\frac{\text{Number of inversions}}{\text{Number of experiments}}$
train_test_split	N/A	35	17.5%
KFold	5	32	16%
ShuffleSplit	2	32	16%
ShuffleSplit	3	36	18%
ShuffleSplit	4	37	18.5%
ShuffleSplit	5	41	20.5%
ShuffleSplit	6	36	18%
ShuffleSplit	7	39	19.5%
ShuffleSplit	8	34	17%
ShuffleSplit	9	31	15.5%
ShuffleSplit	10	31	15.5%

Table 1: Table summarizing the results of the experiments conducted with 500 data samples, a noise with standard deviation 1 in the data generation process, and a noise with standard deviation 0.2 in the noisy estimator, with a constant test size of 20%. The *Number of Inversions* column indicates the number of experiments out of the 200 ran where the noisy estimator outperformed the Bayes estimator.

cases where a suboptimal estimator outperforms the theoretically optimal Bayes estimator. These findings emphasize the importance of carefully designing and interpreting benchmarking studies, as small changes in methodology can lead to misleading conclusions. As the field continues to evolve, it is crucial to develop more robust and standardized benchmarking practices to ensure that comparisons are meaningful and reliable.

References

1. Deng, J. *et al.* ImageNet: A large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition* (2009).
2. Bellemare, M. G. *et al.* The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research* **47,** 253–279 (2013).
3. Moreau, T. *et al.* Benchopt: Reproducible, efficient and collaborative optimization benchmarks. *NeurIPS* (2022).
4. Bouthillier, X. *et al.* Accounting for Variance in Machine Learning Benchmarks. *MLSys* (2021).
5. Sarfraz, M. S. *et al.* Position: Quo Vadis, Unsupervised Time Series Anomaly Detection? *ICML* (2024).
6. Lecun, Y. *et al.* Gradient-based learning applied to document recognition. *Proceedings of the IEEE* (1998).
7. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in Python. *Machine Learning in Python* (2011).
8. Chen, T. & Guestrin, C. XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016).