Original Article

Evaluation of a sampling approach for computationally efficient uncertainty quantification in regression learning models

Valerio Freschi1 · Emanuele Lattanzi1

Received: 7 January 2022 / Accepted: 22 May 2022 / Published online: 11 June 2022 © The Author(s) 2022

Abstract

The capability of effectively quantifying the uncertainty associated to a given prediction is an important task in many applications that range from drug design to autonomous driving, providing valuable information to many downstream decision-making processes. The increasing capacity of novel machine learning models, and the growing amount of data on which these systems are trained poses however significant issues to be addressed. Recent research advocated the need for evaluating learning systems not only according to traditional accuracy metrics but also according to the computational complexity required to design them, toward a perspective of sustainability and inclusivity. In this work, we present an empirical investigation aimed at assessing the impact of uniform sampling on the reduction in computational requirements, the quality of regression, and on its uncertainty quantification. We performed several experiments with recent state-of-the-art methods characterized by statistical guarantees whose performances have been measured according to different metrics for evaluating uncertainty quantification (i.e., coverage and length of prediction intervals) and regression (i.e., errors measures and correlation). Experimental results highlight possible interesting trade-offs between computation time, regression and uncertainty evaluation quality, thus confirming the viability of sampling-based approaches to overcome computational bottlenecks without significantly affecting the quality of predictions.

Keywords Uncertainty quantification · Regression · Sampling · Computational efficiency

1 Introduction

Recent trends in machine learning research witness the increasing importance of quantifying the uncertainty associated to a given prediction task. This is a critical information that results crucial in high stakes applications, transversally to classification and regression. Regression tasks and their relevant use in disparate fields are, in particular, the main focus of this work.

In general, uncertainty quantification plays a pivotal role in enhancing predictions [1–3]. For example, the early assessment of a drug efficacy, clinical survival predictions, autonomous driving, weather forecast or credit default risk are a few, prominent, examples where equipping the machine learning model with the capability of estimating how much it can be (un)certain on its output values allows to gain useful (often crucially important) information for the downstream decision-making. Apart from where mistakes incur large costs, a variety of other applications can in principle benefit from the availability of uncertainty measures, from model-based reinforcement learning to Bayesian optimization (see for instance [4] and/or references therein). There are many ways of performing uncertainty quantification. A typical one involves the estimation of lower and upper bounds defining a prediction interval within which the response variable takes value with high probability. In this framework, prediction intervals should provide an adequate coverage (i.e., the prediction interval contains the true target response variable with probability of \((1 - \alpha)\), for \(\alpha \in (0, 1)\)) and should be of length as small as possible, in order to ensure meaningful and informative prediction. Moreover, avoiding strong assumptions on the statistical distributions and also

---

1 Department of Pure and Applied Sciences, University of Urbino, Piazza Repubblica, 13, 61029 Urbino, Italy
devising methods adaptive to possible heteroscedasticity (by adjusting the length of intervals on the basis of data local variability) are preferable [2].

At the same time, efficiency issues arise when training learning models such, for instance, deep neural networks, that encompass huge numbers (up to millions) of parameters to be optimised. This often could impair the engagement in deep learning research from economically disadvantaged research institutions/centers or scholars, and it also represents a challenge for sustainable computing. The design, training, tuning and deployment of high-accuracy, large-scale learning systems also result into meaningful carbon footprint [5]. As a matter of fact, computational resources required for state-of-the-art solutions (especially in neural network research) have increased of more than 300,000x from 2012 to 2018 [5]. Designing machine learning models that can be more efficiently developed, trained, and also used for inference, can in principle reduce the waste of computational and engineering efforts. Again, since training a single model could require from days to weeks, the search of optimized neural network architectures and hyperparameters results into burdensome computational requirements. For example, as pointed out by Asi et al. [6], the parameter search for obtaining the results reported in some recent research articles, like [7], entailed 750,000 CPU days of computation. Assuming a power consumption in the range of [60 – 100] W, this corresponds to $4 \cdot 10^{12} - 6 \cdot 10^{12}$ J of energy, namely the amount needed (at $10^9$ J per tank of gas) for driving 4000 Toyota Camrys from San Francisco to Los Angeles (380 miles).

To overcome the issues related to these trends, Schwartz et al. advocated the adoption of efficiency as a further evaluation metric, together with accuracy-related metrics traditionally used to assess the quality of a machine learning system, in order to support a more inclusive and greener artificial intelligence [5].

In this article, we aim at pursuing this methodological approach by applying it to the problem of uncertainty quantification (hereafter also denoted as UQ). In particular, we propose to evaluate the effect of uniform sampling on UQ in regression tasks across different benchmark datasets. UQ in regression represents, to the best of our knowledge, a relatively unexplored area of research for what regards joint efficacy-effectiveness performance. On one hand, performing UQ in the case of regression entails a computational effort which is at least difficult (and often worse than) as the underlying regression task alone, hence further exacerbating the problem of how to scale up to large datasets. On the other hand, applying sampling techniques to relieve computational burden for regression UQ needs careful evaluation according to a range of metrics, both pertaining uncertainty (e.g., coverage and interval lengths) and regression accuracy (e.g., mean absolute error and/or root mean square error).

To this purpose, we performed an extensive experimental assessment aimed at carefully evaluating the balancing between the beneficial effects of sampling in terms of training time on some state-of-the-art uncertainty quantification regression algorithms, and the (possibly) negative impact on some key UQ and regression accuracy metrics.

In summary, with this work, we present the following contributions:

1. the use of uniform sampling for computationally efficient uncertainty quantification in regression applications;
2. its application to recent state-of-the-art methods with statistical guarantees/properties;
3. the empirical evaluation of the proposed method along different dimensions (e.g., metrics) and on different dataset;
4. the characterization of the trade-off between time complexity, uncertainty quantification metrics, and regression accuracy metrics.

The remainder of the article is organized as follows: In Sect. 2, we briefly recapitulate significant related work and background; in Sect. 3, we describe the proposed methodology and we detail the performance metrics adopted for our empirical investigation; in Sect. 4, we illustrate the results of the experiments and discuss the main findings; in Sect. 5, we draw some final conclusions.

## 2 Related work

Uncertainty is often represented by means of quantiles that can be used to compute prediction intervals, because of the possibility of modeling distributions without particular parametric assumptions. Quantile regression [8], that allows data-driven estimate of conditional quantile functions, can be used to this purpose. For instance, a conditional quantile function at 2.5 and 97.5% can be used to construct prediction intervals with a 95% coverage. Although this approach has been proved to be suitable for heteroscedastic data processing, formal validity of estimated intervals has been demonstrated to be valid only under specific mathematical conditions (i.e., for given regularity and asymptotic conditions, and only for specific models) [2].

Another line of research that has recently inspired several works is represented by the so-called conformal prediction (CP), a method for the construction of prediction intervals with nonasymptotic, distribution-free coverage
guarantee. The crux of CP methods is that regression models are as usually fitted on the training set, while the obtained residuals are used, with a hold-out validation set for the quantification of future predictions uncertainties [2, 9, 10].

An approach based on the combination of conformal prediction and quantile regression has been recently proposed by Romano et al. [2] in their work. This method, named conformalized quantile regression (CQR for short), integrates both techniques, inheriting finite sample and distribution-free viability (from CP), and flexibility in the construction of prediction interval with different algorithms (from quantile regression). In CQR, the training data are divided into two non-overlapping datasets: a training set and a calibration set. The proper training set is used to train two quantile regressors, that provide first-cut estimates of the prediction interval bounds. Then, this estimate is corrected through the conformalization based on data from the calibration set, allowing to guarantee coverage requirements independently of the underlying quantile regression estimator adopted. Since in CQR prediction intervals are calibrated by means of conditional quantile regression (instead of conditional mean regression as in previous CP studies), the resulting method provides adaptivity to heteroscedastic data.

Other works have recently explored the possibility of constructing prediction intervals by proper modification of the jackknife statistical method, a leave-one-out resampling procedure that has been introduced with the purpose of estimating variability in statistical models [11, 12]; jackknife-based estimate of prediction intervals consists in deriving intervals centered on the output predicted value of the regression task, with width computed from the quantiles of residuals obtained in a leave-one-out processing. The authors of [13] improved this technique by proposing to also make use of leave-one-out predictions at test points in order to gauge the variability of regression. The resulting approach, termed jackknife+, enables distribution-free coverage guarantees under the hypothesis of exchangeable training samples (contrarily to the original jackknife method).

Alaa and van der Schaar finally proposed another variation on theme of jackknife methods, aimed at computing confidence intervals from a frequentist point of view [14]. Particularly, they put forward the use of a technique developed in the field of robust statistics and variational calculus (influence functions), for estimating parameters of learning models that are trained in a leave-one-out fashion, in order to avoid to re-train the model on each held out training point. Despite this workaround, the computational bottleneck inherent to exhaustive leave-one-out training methods is significant, and the scalability of the method remains an issue to be addressed as pointed out by the same authors [14].

Finally, scholars have also investigated the relation between the performance of learning systems in solving specific tasks and the amount of resources needed to solve them, an information that can be summarized by means of so-called learning curves. Performance is usually measured by means of specific accuracy metrics, while resource usage can be the number of examples a model has been trained on, or the time spent while training. In this framework, learning curves contain therefore valuable information for decision-making in many scenarios. For instance, they can be used to evaluate the economical convenience of obtaining more labeled examples, to decide strategies aimed at minimizing training times or preventing overfitting (i.e., Early Stopping techniques), or when to stop evaluating a model (a candidate among a set of possible alternatives) when it is reasonably certain that it could not achieve the performance of other solutions (i.e., Early Discarding techniques in model selection) [15].

Indeed, learning curves are built for increasing sizes of the training set and are terminated when the performance does not significantly change between specific points considered as anchors. Methods differ on how learning curves are derived and on the criteria adopted for early identification of saturation points (points where the performance curve converge). For instance, Provost et al. [16] defined a progressive sampling schedule, where samples are increasingly added according to a geometric progression growth. They also proposed to identify convergence by estimating a linear regression line on a given number of additional points after the latest sample size; if the slope of the regression line, which is tangent to the learning curve and commonly decreases, becomes sufficiently close to zero, convergence is claimed as detected.

Other works have applied various techniques to estimate performance metrics when switching from smaller datasets regimes to larger ones. Particularly, linear, logarithmic, exponential or power law parametric models have been used to model the behavior of different learning systems, from decision trees to statistical machine translation [17, 18]. In [18], a probabilistic model obtained as a weighted combination of a given number of these basic models has been proposed for hyperparameter optimization, to automatically terminate the training of models that are expected to lead to poor performance during the steps of stochastic gradient descent.
3 Methods

To evaluate the impact of uniform sampling on UQ in regression tasks, we chose CQR as our reference UQ model, because of its theoretical properties and state-of-the-art practical performance. Another interesting feature of CQR is to be model-agnostic, in that it allows to wrap different quantile regression methods, thus allowing to easily include in the assessment different types of machine learning models.

We therefore applied the following processing flow: We selected four datasets commonly used as benchmarks in many machine learning experimental assessments. For each dataset, we sampled the training sets according to different sampling ratio and we trained accordingly a range of models for quantile regression within the CQR framework to compute pointwise prediction intervals. We measured the running time taken to perform regression and quantifying uncertainty, and finally, we computed a set of specific UQ metrics and regression accuracy metrics. In the following, we illustrate the details of the machine learning models tested, of the datasets used and of the evaluation metrics adopted.

3.1 Machine learning models

For what concerns conformalized quantile regression, following the work of Romano et al. [2], we evaluated:

- Ridge: a ridge regression model (with regularization parameter tuned by means of cross-validation);
- Ridge-L: a locally adaptive conformal prediction exploiting the ridge regression algorithm; briefly, this is another approach aimed at adapting conformal prediction to heteroskedastic data. In particular, the local ridge regression method makes use of nearest neighbor regression as the estimator of the conditional mean absolute deviation, with the number of neighbors set to 11 (see [2] for details). From an efficiency point of view, this method entails the sequential fit of two functions on the training set, resulting into higher computational complexity with respect to standard conformal prediction.
- Net-L: a feedforward neural network, with locally adaptive conformal prediction (described above). For what concerns the architecture, this network has three fully connected layers: an input layer (with number of units equal to the dimensionality of input data), a hidden layer (with 64 neurons) and a (linear) output layer in charge of returning the value of the response variable; ReLu was used for all nonlinear functions; the model has been trained by minimization of a quadratic loss function using Adam [19] as stochastic optimization algorithm, with minibatches of size 64, weight decay parameter of $10^{-6}$, and learning rate of $5 \cdot 10^{-4}$; dropout [20] (with probability of retaining a hidden neuron set to 0.1) has been used for regularization and early stopping (with an upper bound of 1000 epochs) to avoid overfitting. This configuration is the same proposed by authors of [2] in their work.
- CQR net: the conformalized quantile regression method presented in [2], with neural networks as quantile regression models. The architecture of the neural network is kept identical to the one introduced so far, but the output is represented by the lower and upper conditional quantiles. Even the training algorithm is the same, albeit the cost function is the pinball loss instead of the quadratic loss.

Ridge regression represents a fast baseline method; while neural networks and CQR models have been shown to be relevant solutions for the same types of problems, as well as local methods, at the cost of higher computational requirements [2]. We also included another widely used, flexible and fast method, i.e., gradient boosting to further extend the spectrum of our experimental investigation:

- GradBoost: a gradient boosting model, for which a decision tree is chosen as a base prediction model, and an ensemble of models of the same type are generated accordingly. The final gradient boosting model is built iteratively by optimizing a differentiable loss function [21]. We used the implementation provided with the Scikit-learn distribution [22] (currently stable version 0.24) with quantile loss and default parameter settings: learning rate = 0.1, number of estimators = 100, Friedman mean squared error function as criterion to measure the quality of a split.

3.2 Datasets

Experiments have been carried out on four datasets that are frequently used in scientific literature as benchmarks for regression, namely:

- bio: a dataset regarding the physicochemical properties of protein tertiary structure (45730 data points of dimension 9) [23];
- qsar: a drug discovery dataset released from ChEMBL database consisting of 4084 points with 410 features [24, 25], where the regression task consists in the prediction of the protein activity levels in drug candidate.
development applications, a task known as quantitative structure activity relationship (a.k.a. QSAR) learning;

- concrete: a dataset about concrete compressive strength as highly nonlinear function of age and components (1030 points with 9 features) [23];
- winequality: a dataset targeting the modeling of wine quality based on physicochemical tests (4898 points with dimension 12 each) [23].

All feature values have been processed in order to get zero mean and unit variance, and subsequently divided by mean absolute values.

### 3.3 Performance evaluation metrics

Since the aim of our study was the exploration of the trade-off between computational effort, uncertainty quantification and accuracy, we needed a suitable set of metrics to properly quantify model performances along these dimensions. Performance according to computation can be obviously straightforwardly measured by CPU time, while average coverage and length of prediction intervals suitably quantify uncertainty. With respect to accuracy of regression model, a more articulate spectrum of possibilities can be considered; indeed we included, taking as reference the approach of Tran et al. [26], the following metrics:

1. Median Absolute Error (MDAE), which provides good robustness with respect to outliers, hence making it a good candidate for most data;
2. Root Mean Squared Error (RMSE), included because of its sensitivity to outliers, that turns into a good measure of worst case accuracy;
3. Mean Absolute Relative Percent Difference (MARPD);
4. Regression score function (r²);
5. Pearson correlation coefficient (corr).

The last three metrics (MARPD, r², corr) provide normalized accuracy measures, resulting into a convenient interpretability.

For every dataset, 80% of the data points have been used for training and 20% for testing. Regarding split conformal prediction, training and calibration sets have been kept to equal sizes and the nominal miscoverage rate to 0.1.

To assess the impact of different sampling levels, we trained all models on each dataset by uniformly sampling equal sizes and the nominal miscoverage rate to 0.1. Five points.

3. The local variant of the neural network model, Net-L, achieves the 90% nominal coverage, even when the full training dataset is used. GradBoost and Ridge are more stable with respect to variations in the sampling ratio but with longer prediction intervals.

4. Coverage percentage and length of prediction intervals are introduced separately for each dataset.

#### Dataset: bio

In Fig. 1a and b, we reported, respectively, the coverage and average length of the prediction intervals computed by the different models at different sampling ratios for the bio dataset. In these experiments, gradient boosting shows coverage ranging from 88.9% for sampling ratio equal to 0.2 and it progressively (although slightly) increases to 89.7% when no sampling is performed. All other models achieve values around the nominal coverage rate for almost all sampling ratios, apart Ridge, which is slightly lower than 90% when dataset size is kept 20%. Regarding the average length, the better performance is reached by CQR Net. The local variant of the neural network model, Net-L, reaches an average length equal to 1.79 for the minimum sampling rate of 0.2, which progressively reduces to 1.57 when the full training dataset is used. Ridge-L and GradBoost and Ridge are more stable with respect to variations in the sampling ratio but with longer prediction intervals.

#### Dataset: qsar

In the case of the qsar dataset (Fig. 2a and b), we observe that all methods, except gradient boosting, which is in some cases moderately below (e.g., 87.97% for sampling ratio 20%), are positioned around 90%. All models incur a decrease in the length of prediction intervals, although with different steepness; for instance, Ridge-L incurs a sharp reduction (36.4%) from 4.20 (at a sampling ratio equal to 0.2) to 2.67 (obtained for training on the whole available data); conversely CQR Net is more robust with respect to sampling, ranging from 2.89 (20% of available data used) to 2.44 (full data 15.6% reduction).

#### Dataset: concrete

Figure 3a and b refer to the coverage levels and interval lengths on the concrete dataset. GradBoost does not achieve the 90% nominal coverage, even when the full dataset is used. CQR Net shows a moderate reduction at lowest sampling ratio values, and milder oscillations for the other cases. The other models follow similar trends, although with undercoverage at 20% training set size. Considering interval lengths, Ridge-based methods appear to be more robust with respect to sampling, although they...
provide longer confidence intervals, while Net-L and CQR Net are more sensitive to the sampling ratio but provide shorter intervals on all the spectrum of sampling ratios. Coverage and lengths referred to the winequality dataset are finally reported in Fig. 4a and b, from which we may observe that all models are around (or to a small degree, below) the nominal coverage. CQR Net undergoes a worse interval length for sampling rate equal to 20%, while it progressively reduces this gap for increasing training set size. Gradboost appears to be less affected by sampling, while being at the same time the one with lowest length values across (almost) all different sizes of training set.

4.2 Regression accuracy

The performance levels obtained by the different machine learning models as measured according to different regression accuracy metrics are separately presented below for, respectively, all the four dataset of our study.

Dataset: winequality

Coverage and lengths referred to the winequality dataset are finally reported in Fig. 4a and b, from which we may observe that all models are around (or to a small degree, below) the nominal coverage. CQR Net undergoes a worse interval length for sampling rate equal to 20%, while it progressively reduces this gap for increasing training set size. Gradboost appears to be less affected by sampling, while being at the same time the one with lowest length values across (almost) all different sizes of training set.

Fig. 1 Experimental results on bio dataset for different sampling ratios. a Coverage. b Prediction interval length. c Mean Absolute Error. d Root Mean Square Error. e Median Absolute Error. f Mean Absolute Relative Percent Difference. g Regression score function $R^2$. h Pearson correlation. i CPU time

Dataset: bio

From the analysis of Fig. 1c–h, we first observe that CQR Net outperforms all models across the different metrics, followed by Net-L (with the only exception of the correlation metrics where Net-L attains higher values). Both models are in general rather robust with respect to sampling and present a moderate improvement of accuracy with the number of training examples. Ridge and Ridge-L remain virtually constant at different sampling levels and
achieve the same accuracies according to all the different metrics, except for the regression score function and for Pearson correlation, which are mildly improved when the training set size is increased. Gradient boosting obtains better accuracies than ridge-based models, but worse than CQR Net and Net-L, and does not present, as well, appreciable sensitivity with respect to variations in the sampling level.

**Dataset: qsar**

Figure 2c–h illustrates the better performance levels of CQR Net and Net-L. Interestingly, these two models trained at a sampling ratio of 0.4 are able to outperform the other models (ridge-based and gradient boosting) trained on the full dataset. GradBoost is generally better than ridge models, especially at low sampling ratios.

**Dataset: concrete**

In this dataset, as exemplified in Fig. 3c–h, Ridge and Ridge-L are significantly worse than GradBoost, Net-L, and CQR Net. For instance, the mean absolute error of CQR Net at 0.6 of sampling ratio (60% of training set used) is, on average, equal to 0.10 against a value more than double (equal to 0.24) for ridge linear regression.

**Dataset: winequality**

Regression models on this dataset (Fig. 4c–h) show in general more homogenous quality profiles if compared to the other three sets. For instance, if we take into consideration the MARPD, the difference between the model with best accuracy (CQR Net) and the one with the worst (Ridge) results to be moderate: when only the 20% of the dataset is used for training, CQR Net achieves a MARPD of...
10.13, while Ridge of 10.16; when the models are trained on all available data, the result for neural nets with conformalized quantile regression is equal to 8.58 and equal to 10.13 for ridge regression.

### 4.3 Training algorithms timing

CPU time required for training the machine learning models has been measured under different sampling ratios and datasets. Experimental results have been reported in Figs. 1i, 2i, 3i, 4i. All experiments have been executed on an Intel Core i7 1.2GHz processor with 16GB RAM, running the macOS operating system.

Ridge-based regression required the lowest computational effort: For instance, if we consider the largest of the four benchmarks (the bio dataset), we can see (Fig. 1i) that training the model on the whole training set necessitated about 37.2 msec for Ridge, and 725.9 msec for Ridge-L. Conversely, CQR Net resulted the most demanding, with about 40 minutes of CPU time, followed by Net-L with 38 minutes.

By aggressively sampling the training set, the time needed to train models reduced to 9.5 minutes for CQR Net, 7.1 minutes for Net-L, 16.6 msec for Ridge, and 189.5 msec for Ridge-L. The performance of GradBoost is also competitive in terms of computational requirements, since it required 6.41 sec when trained on 20% of the available training data and 34.5 sec on the whole training bio dataset.
Timings for training models on the remaining datasets (qsar, concrete, and winequality) present similar relative trends as apparent from the analysis of Figs. 2i, 3i, 4i.

4.4 Discussion

The results of experiments presented previously are multi-faceted. Nonetheless, some insights can be gained and some considerations drawn from their analysis.

First of all, it can be highlighted that uncertainty quantification adds another dimension to the assessment of regression learning. Models that are usually evaluated according only to accuracy measures may no longer be effective (at least to the same extent) if one takes into account coverage and/or length of prediction intervals as further parameter of evaluation. Our experiments confirm indeed that not all methods (e.g., ridge regression or gradient boosting) guarantee a given nominal coverage and, in particular, they output prediction intervals with different lengths. In this sense, the results of our investigation confirm the superiority of conformal prediction methods (i.e., CQR Net and locally adaptive Net-L) that provide formal statistical guarantees on coverage and improved interval lengths. On the other hand, the time complexity required for training given models, such as neural networks, motivates the need of taking into account temporal dimension as a key evaluation parameter.

A viable strategy to mitigate the computational burden of training algorithms is represented by sampling strategies. According to our experiments, this obviously reduces the computational requirements and, in general, does not impact dramatically some metrics for given models, which

Fig. 4 Experimental results on winequality dataset for different sampling ratios. a Coverage. b Prediction interval length. c Mean Absolute Error. d Root Mean Square Error. e Median Absolute Error. f Mean Absolute Relative Percent Difference. g Regression score function R2. h Pearson correlation. i CPU time.
represents a less trivial result. For instance, training on bio/ qsar datasets at a sampling ratio of 0.2 required, respectively, 9.5/1.2 minutes for CQR Net and 7.1/0.7 minutes for Net-L. Although it took less than a second for training Ridge-L on the full version of these datasets, CQR Net and Net-L trained on 20% of the dataset largely outperformed it, both according to uncertainty quantification metric (e.g., prediction intervals lengths) and to regression accuracy, with a reasonable execution time.

These findings can be integrated and complemented by other works targeting the study of learning curves. For instance, adopting a progressive sampling strategy (like the one proposed in [16]), where uncertainties are properly quantified by means of prediction intervals and their coverage, could in principle enable an algorithmic approach that minimizes the effort spent during training while maximizing performance (i.e., smaller interval lengths at fixed coverage probabilities). This would provide designers a principled approach to explore performance-accuracy trade-offs in the design of learning models.

In fact, through proper extrapolation of the information provided by learning curves (for instance using parametric models [18]), and by detecting the convergence to plateaus of performance [16], it should be possible to take early decisions during training that could enable savings in computation times without decreasing accuracies. As a future work, it would also be interesting to investigate combined metrics that could convey different type of performance measures into a single one, to be used in a subsequent learning curve study.

5 Conclusions

Uncertainty quantification is key to the design of reliable learning systems since it enables to complement predictions with a measurement of the how much a given model is (un)certain regarding the inference made. This is a central issue in high stakes applications as health care diagnosis, drug design or autonomous driving, but it can be considered in general an effective means to improve the effectiveness of many decision-making processes.

Recent developments in machine learning techniques (in particular those based on neural networks) complemented by the increasing availability of datasets of significant size have raised the issue of how to effectively cope with the computational load to sustained for training state of the art models. A recent trend in scientific literature advocates a wider view of the design and evaluation cycle of a learning system, toward a more inclusive and greener artificial intelligence.

In this article, we described the results of an experimental study that we conducted to establish how uniform sampling applied to training data could be a feasible approach to the problem. In particular, we studied the effect of sampling on i) uncertainty quantification, as measured by prediction intervals coverage and length; ii) regression quality, as measured by several error and correlation metrics; iii) time required to train the model.

The results point out that sampling can be fine-tuned to achieve useful trade-offs between the three dimensions of computational requirements, uncertainty quantification capability and regression accuracy. For example, sampling enables a 4x speedup in training time for conformalized regression with neural networks, hence potentially resolving the computation bottleneck without negatively affecting performance on UQ and regression quality with respect to standard baseline methods like ridge regression, even if these fully exploit available data.

Acknowledgements This project has received funding from the Department of Pure and Applied Sciences (DiSpEA), University of Urbino under institutional research Grant 2020-2022.

Funding Open access funding provided by Università degli Studi di Urbino Carlo Bo within the CRUI-CARE Agreement.

Declarations

Conflict of interest The authors declare that they have no conflicts of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Gal Y, Ghahramani Z (2016) Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: International Conference on Machine Learning, pp. 1050–1059. PMLR
2. Romano Y, Patterson E, Candes E (2019) Conformalized Quantile Regression. In: Wallach, H., Larochelle, H., Beygelzimer, A., d’ Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 32
3. Van Mole P, Verbelen T, Vankeirsbilck B, De Vylder J, Diriex B, Kimpe T, Simoens P, Dhoecht B (2021) Leveraging the Bhattacharyya coefficient for uncertainty quantification in deep neural networks. Neural Comput Appl 33:10259–10275
4. Chung Y, Neiswanger W, Char I, Schneider J (2021) Beyond Pinball Loss: Quantile Methods for Calibrated Uncertainty
Quantification. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P.S., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems, vol. 34, pp. 10971–10984
5. Schwartz R, Dodge J, Smith NA, Etzioni O (2020) Green AI. Commun ACM 63(12):54–63
6. Asi H, Duchi JC (2019) The importance of better models in stochastic optimization. Proc Natl Acad Sci 116(46):22924–22930
7. Collins J, Sohl-Dickstein J, Sussillo D (2016) Capacity and trainability in recurrent neural networks. arXiv preprint arXiv:1611.09913
8. Koenker R, Bassett Jr G (1978) Regression quantiles. Econometrica: J Econom Soc, 33–50
9. Papadopoulos H (2008) Inductive conformal prediction: Theory and application to neural networks. In: Tools in Artificial Intelligence. InTech, ???
10. Vovk V (2012) Conditional validity of inductive conformal predictors. In: Asian Conference on Machine Learning, PMLR, pp. 475–490
11. Miller RG (1974) The jackknife—a review. Biometrics 61(1):1–15
12. Efron B (1992) Bootstrap methods: another look at the jackknife. Breakthr Stat. Springer, London, pp 569–593
13. Barber RF, Candes EJ, Ramdas A, Tibshirani RJ (2021) Predictive inference with the jackknife+. Ann Stat 49(1):486–507
14. Alaa A, Van Der Schaar M (2020) Discriminative jackknife: Quantifying uncertainty in deep learning via higher-order influence functions. In: International Conference on Machine Learning, pp. 165–174. PMLR
15. Mohr F, van Rijn JN (2022) Learning Curves for Decision Making in Supervised Machine Learning—A Survey. arXiv preprint arXiv:2201.12150
16. Provost F, Jensen D, Oates T (1999) Efficient Progressive Sampling. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 23–32
17. Mukherjee S, Tamayo P, Rogers S, Ritkin R, Engle A, Campbell C, Golub TR, Mesirov JP (2003) Estimating dataset size requirements for classifying DNA microarray data. J comput biology 10(2):119–142
18. Domhan T, Springenberg JT, Hutter F (2015) Speeding up Automatic Hyperparameter Optimization of Deep Neural Networks by Extrapolation of Learning Curves. In: Twenty-fourth International Joint Conference on Artificial Intelligence
19. Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980
20. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J mach learn res 15(1):1929–1958
21. Friedman JH (2002) Stochastic gradient boosting. Comput stat data anal 38(4):367–378
22. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830
23. Dua D, Graff C (2017) UCI Machine Learning Repository. http://archive.ics.uci.edu/ml
24. Olier I, Sadawi N, Bickerton GR, Vanschooren J, Grosan C, Soldatova L, King RD (2018) Meta-QSAR: a large-scale application of meta-learning to drug design and discovery. Mach Learn 107(1):285–311
25. Diakonikolas I, Kamath G, Kane D, Li J, Steinhardt J, Stewart A (2019) Sever: A Robust Meta-Algorithm for Stochastic Optimization. In: International Conference on Machine Learning, pp. 1596–1606. PMLR
26. Tran K, Neiswanger W, Yoon J, Zhang Q, Xing E, Ulissi ZW (2020) Methods for comparing uncertainty quantifications for material property predictions. Mach Learn Sci Technol 1(2):025006

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.