Published in Image Processing On Line on 2026–06–00.
Submitted on 2023–10–11, accepted on 2026–03–31.
ISSN 2105–1232 © 2026 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2026.514

An Active Learning Method Based on Bagging and Applied to Surrogate Modelling

Nomena Andrianarisoa, Matthieu Ancellin, Vincent Laurent

Mews Labs, Cachan, France
Corresponding author: vincent.laurent@mews-labs.com

Communicated by Gabriele Facciolo *Demo edited by* Vincent Laurent and Andrianarisoa Nomena

Abstract

In this work, we study active learning techniques to highlight their strengths and limits in the context of surrogate modeling. We implemented an iterative algorithm that uses active sampling methods to approximate known functions. Through a series of experiments, we demonstrate the effectiveness of active learning in improving surrogate models by wisely selecting informative data points. Furthermore, we examine cases where active learning may face challenges, particularly when balancing exploration and exploitation. This study provides information on the improvement of the efficiency and accuracy of surrogate models in practical applications.

Source Code

The reviewed source code and documentation for this algorithm are available from the web page of the article[1]. Usage instructions are included in the `README` file of the archive.

Keywords: active sampling; bootstrap approach; regression

1 Introduction

In today's scientific and engineering domains, a multitude of physical phenomena are being numerically modeled at various scales, ranging from macroscopic to microscopic levels. For example, phenomena such as material fatigue, where the structural integrity of components degrades over time due to repeated loading, require simulations that cover a wide range of variables such as stress, strain, and environmental conditions. The computational requirements of these simulations are amplified by the sheer number of input parameters needed to accurately describe these complex problems. As a result, conducting exhaustive simulations becomes both computationally expensive and time consuming.

For these reasons, the development of surrogate models [2] can serve as efficient substitutes, particularly for modeling small-scale phenomena. Surrogate models offer a practical solution by capturing

[1] https://doi.org/10.5201/ipol.2026.514

the essential features of the system while reducing the computational complexity associated with detailed simulations. To address the computational challenges posed by these complex simulations, the integration of active learning techniques in the development of surrogate models has emerged as an interesting approach.

Active learning [13] enables the surrogate model to selectively acquire new data points, optimizing the trade-off between computational resources and model accuracy. By actively querying the most informative data points, the surrogate model can effectively reduce the number of required simulations while maintaining a high level of accuracy. This iterative process of actively selecting informative samples not only accelerates the construction of the surrogate model but also allows for adaptability and refinement as additional simulations are performed. Therefore, the integration of active learning methodologies into the development of surrogate models [10] holds great promise for improving computational efficiency in the modeling of multiscale physical phenomena.

The work presented in this demonstrator aims to illustrate the strengths of active learning for surrogate models by using it to approximate known analytical functions.

2 Method Description

As stated above, active learning can significantly reduce the number of training points needed by selectively choosing the most informative ones. It also helps to concentrate on the regions of the input space that are the most relevant for predicting continuous values, leading to faster and more effective improvement in model performance. To do so, active sampling methods [6] can be used. Active sampling methods can involve techniques in which training points are selected in an iterative manner. By iteratively selecting points, the model can, for instance, focus on regions of the input space where it has high variability in its predictions. This allows the model to adapt and refine its predictions in these regions, leading to improved performance over time. The active method is based on:

- a method to estimate the uncertainty of the model. By computing several variants of the same model with slightly different training data, the bagging method provides an easy way to estimate the sensitivity of the model with respect to the training data.

- a sampling method to select new data points to be integrated into the training dataset based on the uncertainty of the model. Two variants are presented below.

2.1 General Structure of the Method

As briefly explained above, the algorithm is an iterative method, described in Figure 1.

The core of this algorithm is to select new points based on their standard deviation to improve the accuracy of the model. By actively sampling around regions of high uncertainty indicated by the standard deviation, the method aims to refine the model's predictions and achieve better approximation of the target functions.

Step 1: Creation of the initial training database. First, an initial set of points is needed to fit the first model. To do so, `n_initial` training points are selected uniformly at random according to the approximate limits of the function f. Once this initial set of points is defined, we evaluate the function f for each point of this set and the initial training database is defined as a set of pairs $D_0 = ((x_1, f(x_1)) \ldots, (x_{n_{\text{initial}}}, f(x_{n_{\text{initial}}}))$.

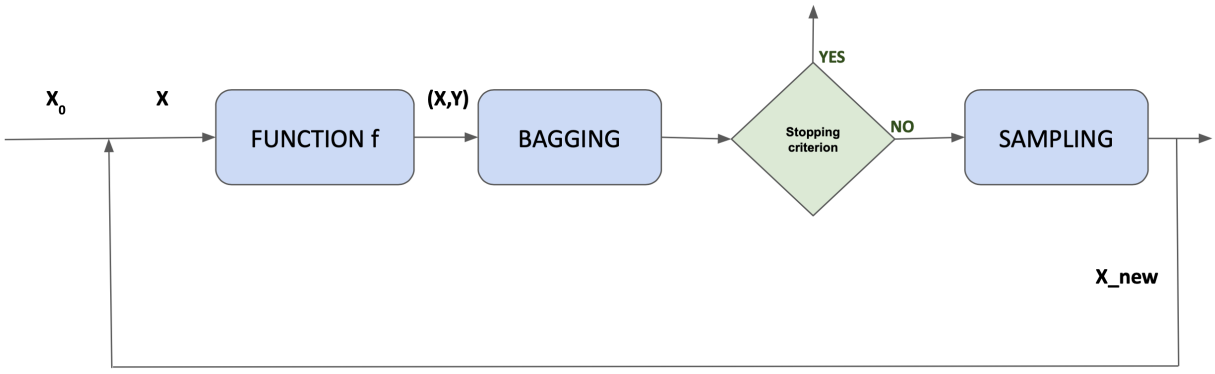

Figure 1: General structure of the algorithm.

Step 2: Model fitting using bagging. Now that we have a training database, we can fit a model using the bagging method on it to approximate the function f. The bagging method [3] is an ensemble technique widely used in machine learning to improve the accuracy of predictive models. The term "bagging", which means bootstrap aggregation, comes from the combination of two fundamental concepts: bootstrap resampling and aggregation. The bagging method uses bootstrap resampling to generate multiple training datasets from a single dataset. Bootstrap resampling is a statistical technique that involves creating random subsets with replacement from the original dataset. Thus, each sample can contain repeated instances, and some instances may be absent. The number of instances in each sample is equal to the size of the original dataset. Once the bootstrap datasets have been generated, a regression model is trained on each bootstrap sample. Each model is trained independently, encouraging diversity in predictions. The predictions of all models are then aggregated to obtain a final prediction. In regression, the most common method of aggregation involves taking the average of the predictions of all models.

Step 3: Choice of new points. Now that we have a model trained on the initial training database, we want to improve the model by adding new points to the training database. To do so, we use active sampling methods [13] based on the standard deviation of the prediction of the models from bagging.

If we denote the data set $\mathcal{D}$ composed of n pairs of observations $(X_i, Y_i)_{i\in[1,n]}$, the variance of the model $\hat{f}_{\mathcal{D}}$ with respect to $\mathcal{D}$ at a given location x can be written as [8]

$$\mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)^2 - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)]^2]\,,$$

where $\mathbb{E}_D$ denotes the expectation over the training dataset. In practice, a natural estimate of this variance is given by

$$\hat{\sigma}^2(x) = \frac{1}{k}\sum_{j=1}^{k}\left(\hat{f}_j(x)\right)^2 - \left(\frac{1}{k}\sum_{j=1}^{k}\hat{f}_j(x)\right)^2,$$

where the functions $\hat{f}_j$ are k different estimates of the function f trained by the bagging method discussed above. A larger standard deviation $\hat{\sigma}(x)$ indicates a higher level of uncertainty in the approximation of f at point x, suggesting that it is appropriate to add $(x, f(x))$ to the dataset to improve the model.

Two variants have been implemented to sample new points according to the standard deviation defined above. Both variants depend on the bounds of the function, the number of new points to be added to the dataset and the standard deviation $\hat{\sigma}$ of the model as defined above.

- The first variant is to *look for maxima of* $\hat{\sigma}$ with a naive Monte-Carlo approach (called `query_max_rand` in the code). A large number of random candidate points are sampled uniformly in the domain of definition of f. The variance of the model predictions $\hat{\sigma}$ is then evaluated at each candidate point. If m new points are required, then the m candidate points with the highest value of $\hat{\sigma}$ are selected.

- The second variant is to *sample new points using* $\hat{\sigma}$ *as a probability density function* (called `query_pdf_rand` in the code). Random candidate points $(x_i)_{i\in[1,N]}$ are sampled uniformly in the domain of definition of the function f and the variance of the model predictions at each point $\hat{\sigma}(x_i)$ is evaluated. Then some of the candidate points are randomly selected among all the candidates using the discrete probability density $p_i = (\hat{\sigma}^2(x_i) + \varepsilon)/\sum_j(\hat{\sigma}^2(x_j) + \varepsilon)$ with $\varepsilon > 0$ to ensure non-zero probabilities.

The latter method is expected to better cover the domain by selecting points not only where $\hat{\sigma}$ is the highest, but also at other locations where $\hat{\sigma}$ is moderately high.

Whatever variant is used to select points with high model uncertainty, only a fraction of the new samples are generated this way. Other samples are selected uniformly in the definition domain of the function. We call *active ratio* the fraction of points that are sampled using $\hat{\sigma}$. The sensitivity of the results with respect to the active ratio is discussed empirically below.

Once all the new points are selected, we evaluate the function f for each point in the set D_{new}. Then, we add these new data points to the training database. We repeat steps 2 and 3 until we reach the stopping criterion. The method is described in Algorithm 1.

2.2 Stopping Criterion

Every iterative method requires a stopping criterion to determine when the method has converged or reached a desired level of accuracy. The convergence of our method can be detected by a convergence criterion on the models of the form [11]

$$\|\tilde{f}_k - \tilde{f}_{k-1}\| < \epsilon,$$

where k represents the current iteration and $\|.\|$ a suitable norm.

However, our implementation relies only on a maximum number of samples `n_full_data` to stop the iterations after a maximum number of iterations

$$\text{iter}_{max} = \left\lfloor \frac{\texttt{n\_full\_data} - \texttt{n\_initial}}{\texttt{n\_new}} \right\rfloor$$

where `n_initial` is the number of points in the initial training database and `n_new` is the number of new points we want to add at each iteration. The same algorithm is repeated several times to take into account the random sampling of new points, and it was more convenient to have the same number of iterations for each repetition, which is why early-stopping methods have not been used. In our demonstrator, suitable values of `n_full_data` and `n_initial` have been preselected for each function f to be approximated.

3 Demo Implementation and Illustrative Examples

We implemented the method described above. In our case, we want to approximate analytical functions in $\mathbb{R}$ or $\mathbb{R}^2$ using active learning in the sampling process based on the standard deviation.

Algorithm 1: Model fitting using active sampling

Input : function f to approximate, bounds of the function
Output : $\tilde{f}$ the approximation of the function f, `database`

```
iter = 1
D_0 the initial set of points of the domain of definition of f
for v in D_0 do
    Evaluate function f to obtain the corresponding target value
    Add (v, f(v)) to database
while iter < iter_max do
    Fit model on database using the bagging method
    Generate n_new points D_new using an active sampling method based on standard
     deviation
    for v in D_new do
        Evaluate function f to obtain the corresponding target value
        Add (v, f(v)) to database
    iter = iter + 1
```

To that end, we use the `BaggingRegressor` [4] from the Python library `scikit-learn`[2]. This regressor is based on the bagging method. In our tests, the seed is fixed so that the reproducibility of the results is ensured. The `BaggingRegressor` allows the use of different types of base estimator when it predicts a model on each subset. In the demo, four regressors from the `scikit-learn` library can be selected: `DecisionTreeRegressor` [5], `ExtraTreesRegressor` [9], `KernelRidge` [12] and `svm.SVR` [7]. The demo also allows the user to set the value of the active ratio. For this paper values of 0.6 and 1 have been selected arbitrarily.

The iterative algorithm is repeated five times. Throughout the iterations, the relative L_2 error between the prediction of the model and the true values of the function is calculated on a test database of 500 points as an approximation of

$$L_2(f, \tilde{f}) = \frac{\|f - \tilde{f}\|_2}{\|f\|_2}$$

where f is the exact (nonzero) function and $\tilde{f}$ is its approximation. Such a large test dataset might not be feasible for real problems, but it is useful in our toy problem to analyze the convergence of the training and compare the efficiency of the active sampling approach compared with random sampling.

3.1 Computation Time and Convergence Analysis

In this section, we compare the performance of two different estimators, namely `DecisionTreeRegressor` and `KernelRidge`, using the same input dataset.

In the following case, we try to approximate the *himmelblau* function [1] defined on $\mathbb{R}^2$ with values in $\mathbb{R}$, adding 10 new points at each iteration using the `query_max_rand` sampling function with an active ratio of 0.6.

[2] https://scikit-learn.org/stable/

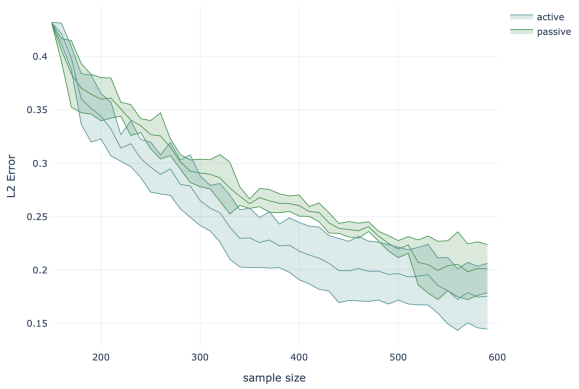

Figure 2: L_2 error comparison between active (in blue) and passive (in green) approaches with `DecisionTreeRegressor`

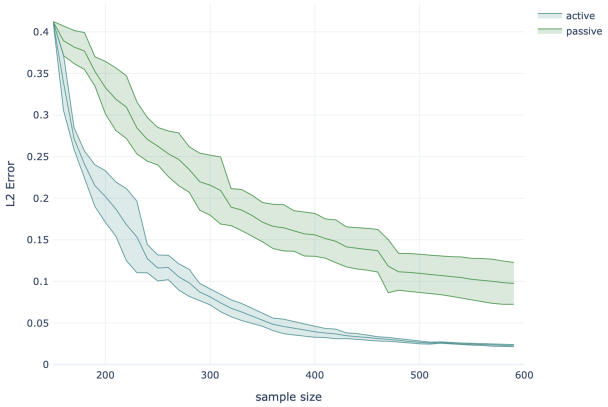

Figure 3: L_2 error comparison between active (in blue) and passive (in green) approaches with `KernelRidge`

The curves show the average relative error L_2 as a function of the number of points in the training dataset. The standard deviation of this error represents the variability across the five independent repetitions of the training process.

For the same input dataset, we notice that the `DecisionTreeRegressor` estimator exhibits faster computation times compared to the `KernelRidge` estimator (5 seconds for `DecisionTreeRegressor` versus 46 seconds for `KernelRidge`). The `KernelRidge` estimator demonstrates higher accuracy, with a mean final relative L_2 error of 0.023, while it is 0.175 for `DecisionTreeRegressor`. These trends are expected given the underlying algorithms and computational complexity of these models.

The results clearly demonstrate that active sampling outperforms random sampling in the majority of cases. In particular, the convergence of the L_2 error is faster in the active sampling approach, represented by the blue curve, compared to passive random sampling, illustrated by the green curve, as shown in Figures 2 and 3.

The accelerated convergence in active sampling is due to its selection of informative data points, which guide the model towards areas of higher uncertainty, leading to quicker learning and model refinement. In addition, active sampling yields higher accuracy, as illustrated by the smaller final L_2 error, focusing on regions of the input space that significantly influence the model's predictions. These results highlight the advantages of active sampling in improving the efficiency and accuracy of function approximation tasks.

3.2 Interaction Between Exploration and Exploitation of the Input Space

Contrary to initial expectations, it has been observed that active sampling may not significantly outperform random sampling and may even be less effective than random sampling, as illustrated in Figures 4 and 5. These results were obtained by approximating a one-dimensional function $f : x \mapsto x^5 \sin(10\pi x)$, adding 10 new points at each iteration using the `query_max_rand` sampling function with an active ratio of 1.0.

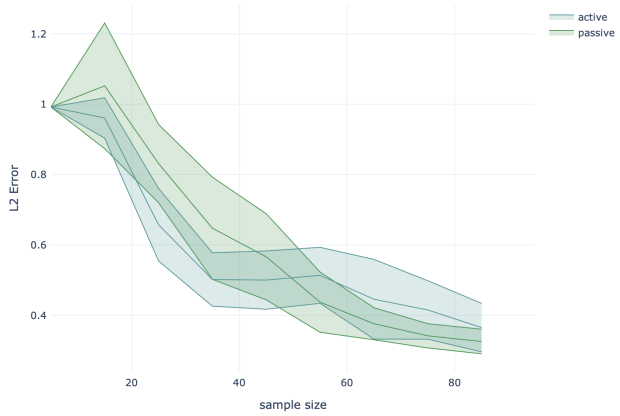

Figure 4: L_2 error comparison between active (in blue) and passive (in green) approaches with `ExtraTreesRegressor`

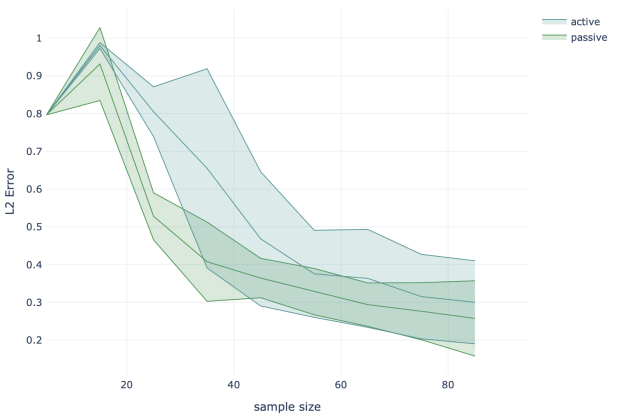

Figure 5: L_2 error comparison between active (in blue) and passive (in green) approaches with `DecisionTreeRegressor`

This phenomenon is caused by the fact that the sampling process focuses mainly on specific regions of the domain, leading to regions without training samples. The procedure may become stuck when the model struggles to converge to the correct value. This situation often arises when the estimator exhibits bias, leading to deviations from the true function. The presence of bias in the estimator can jeopardize the model's ability to effectively adapt and refine its predictions, hindering the convergence process, and potentially limiting the overall performance of the active learning approach.

This highlights the significance of the active sampling ratio, which in this case is set to 1. To foster effective model refinement, it is crucial to target regions with large uncertainties, indicated by the high standard deviation. However, it is equally important to consider domain exploration to ensure a complete understanding of the underlying data distribution.

To gain a better understanding of this phenomenon, it is interesting to visualize both the true and predicted functions, along with the final distribution resulting from the active sampling process. In the following example, we again try to approximate $f(x) = x^5 \sin(10\pi x)$ adding 5 new points at each iteration using the `query_max_rand` sampling function and the `svm.SVR()` base estimator. By comparing simulations that differ only in the active sampling ratio, we can effectively assess the impact of targeting specific regions of interest.

The visual representation of the true and predicted functions (Figures 6 and 7) provides a comprehensive assessment of the accuracy of the model and its ability to approximate the true function. At the same time, visualizing the distribution resulting from the active sampling process (Figures 8 and 9) allows us to observe how the sampling strategy targets regions with high uncertainty and how it influences the model's learning dynamics.

For the case where the active ratio is set to 1, Figure 6 reveals that the approximation of the function at the center of the domain is poor, showing a noticeable bump. The corresponding distribution shown in Figure 8 illustrates that no points were drawn in the corresponding region, resulting in the absence of training points.

In contrast, for an active ratio of 0.6, Figure 7 shows a highly accurate approximation of the true function. In the distribution depicted in Figure 9, we observe that the entire domain is covered by the final training database, with a higher concentration of points in regions with significant variations.

This underscores the essential role of exploration in the active sampling process. The absence of exploration, as evidenced by the lack of points in certain regions for an active ratio of 1, can compromise the model's ability to effectively approximate the true function. This illustrates the importance of finding a balance between exploration and exploitation when optimizing the active sampling procedure.

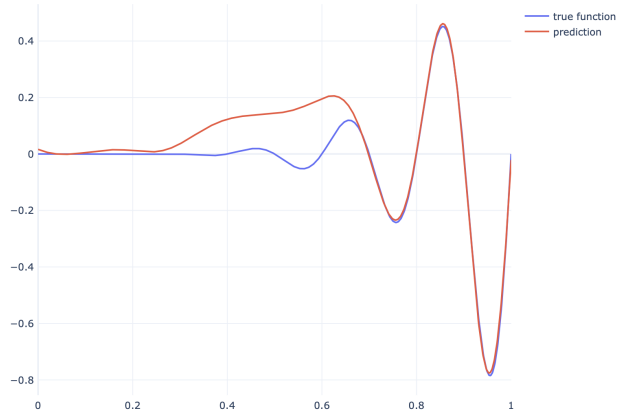

Figure 6: True and predicted functions with ratio = 1.0

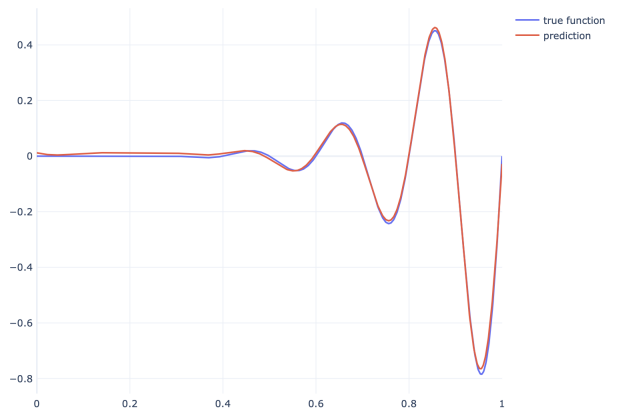

Figure 7: True and predicted functions with ratio = 0.6

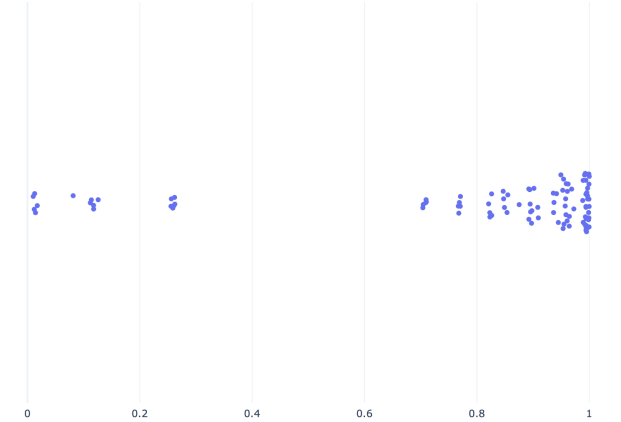

Figure 8: Distribution of the active sampling process with active ratio = 1.0

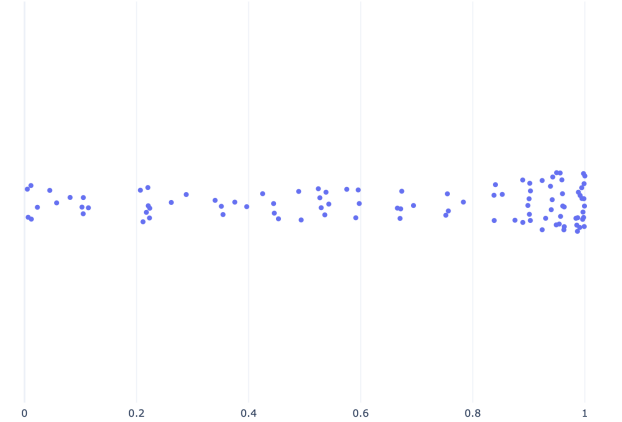

Figure 9: Distribution of the active sampling process with active ratio = 0.6

4 Conclusion

In conclusion, the active learning approach we study is a versatile plug-in method that can be integrated with any estimator. Its effectiveness lies in its ability to intelligently select data points from regions of high uncertainty, which, in certain cases, accelerates model refinement. Using informative data points, the active learning approach optimizes the learning process and enhances the accuracy of the model. This versatility and efficiency make it a valuable tool for function approximation tasks, where selecting the right training data can significantly impact the quality of the final model. However, we have observed cases where active sampling may not be as efficient, highlighting the critical importance of achieving a balance between exploration and exploitation, especially in scenarios with a high active sampling ratio.

Moving forward, exploring the application of this method to higher-dimensional cases beyond one or two input variables is an interesting direction for future work. In addition, we envision the application of this approach to real-world simulations, particularly in cable fatigue analysis, providing valuable insights into fatigue phenomena and improving predictive capabilities in practical engineering applications.

References

[1] N. Andrei, *An Unconstrained Optimization Test Functions Collection*, Advanced Modeling and Optimization, 10 (2008), pp. 147–161.

[2] A. Belkhabbaz, M. Gueguin, F. Hafid, C. Yang, O. Allix, and J.-M. Ghidaglia, *Surrogate Model Based Approach to Predict Fatigue Stress Field in Multi-Stranded Cables*, International Journal of Solids and Structures, 230 (2021), p. 111168, https://doi.org/10.1016/j.ijsolstr.2021.111168.

[3] L. Breiman, *Bagging Predictors*, Machine Learning, 24 (1996), pp. 123–140, https://doi.org/10.1007/BF00058655.

[4] ——, *Pasting Small Votes for Classification in Large Databases and On-Line*, Machine Learning, 36 (1999), pp. 85–103, https://doi.org/10.1023/A:1007563306331.

[5] L. Breiman, J. Friedman, and C. J. Stone, *Classification and Regression Trees*, CRC Press, 1984, https://doi.org/10.1201/9781315139470.

[6] R. Burbidge, J. J. Rowland, and R. D. King, *Active Learning for Regression Based on Query by Committee*, in International Conference on Intelligent Data Engineering and Automated Learning, Springer, 2007, pp. 209–218, https://doi.org/10.1007/978-3-540-77226-2_22.

[7] C.-C. Chang and C.-J. Lin, *LIBSVM: a Library for Support Vector Machines*, ACM Transactions on Intelligent Systems and Technology (TIST), 2 (2011), pp. 1–27, https://doi.org/10.1145/1961189.1961199.

[8] S. Geman, E. Bienenstock, and R. Doursat, *Neural Networks and the Bias/Variance Dilemma*, Neural Computation, 4 (1992), pp. 1–58, https://doi.org/10.1162/neco.1992.4.1.1.

[9] P. Geurts, D. Ernst, and L. Wehenkel, *Extremely Randomized Trees*, Machine Learning, 63 (2006), pp. 3–42, https://doi.org/10.1007/s10994-006-6226-1.

[10] L. Hong, H. Li, and J. Fu, *A Novel Surrogate-Model Based Active Learning Method for Structural Reliability Analysis*, Computer Methods in Applied Mechanics and Engineering, 394 (2022), p. 114835, https://doi.org/10.1016/j.cma.2022.114835.

[11] H. Ishibashi and H. Hino, *Stopping Criterion for Active Learning Based on Error Stability*, ArXiv Preprint ArXiv:2104.01836, (2021), https://doi.org/10.48550/arXiv.2104.01836.

[12] K. P. Murphy, *Machine Learning: a Probabilistic Perspective*, MIT Press, 2012. ISBN 978-0-262-01802-9.

[13] B. Settles, *Active Learning Literature Survey*, Tech. Report TR1648, University of Wisconsin-Madison Department of Computer Sciences, 2009. https://minds.wisconsin.edu/handle/1793/60660.