Geoscientific Model Development Discussions

Geosci. Model Dev. Discuss., referee comment RC1
https://doi.org/10.5194/gmd-2020-445-RC1, 2021

Comment on gmd-2020-445

Anonymous Referee #1

Referee comment on "DecTree v1.0 – chemistry speedup in reactive transport simulations: purely data-driven and physics-based surrogates" by Marco De Lucia and Michael Kühn, Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2020-445-RC1, 2021

This is a well-conceived paper on creating surrogate regressor models to speed up the expensive physics solvers found in geochemical models. The authors present clever feature engineering techniques based on geochemical knowledge and understand their modeling system well by employing a Tweedie distribution to scale their training data. The use of a mass-balance metric to call the original solver once the machine learning (ML) model starts to drift is interesting. They present a balanced view on using machine learning and combining it with physics-informed domain knowledge. The paper fits into the scope of GMD.

Overall the manuscript needs very minor revision but better justifications for the ML models must be made as they currently remain overly-simplistic and potentially misleading.

Suggestions:

~Line 60: "For this reason, we argue that the most sensible choice for a surrogate modelling framework is that of multiple multivariate regression: one multivariate regressor - making use of many or all inputs as predictors - is trained independently for each distinct output variable, while the choice of regressor may vary from variable to variable."

So, each output variable has its own ML model? This seems like it is not well explained, and a reader might believe this to be more computationally expensive. A better framing is that it is difficult to get all output variables to perform well with a single ML model. A citation that shows this here:

Kelp, M. M., Jacob, D. J., Kutz, J. N., Marshall, J. D., & Tessum, C. W. (2020). Toward stable, general machineâ□□learned models of the atmospheric chemical system. Journal of Geophysical Research: Atmospheres, 125, e2020JD032759. https://doi.org/10.1029/2020JD032759

where you can optimize a specific output variable's performance above others (thereby retaining the same input/output dimensionality), or just simply make new ML models for

each output. This shows that one ML model is not effective in capturing all variables and that multiple specialized models must be made.

~Line 66: "In praxis, the CPU-time, the user interactions and the overall skills required for optimally training complex regressors cannot be underestimated and may prove overwhelming for a geoscientist. The whole process of hyperparameter tuning, required by most advanced machine learning algorithms, while being an active area of development and research, is still hardly fully automatable."

I do not like this reasoning too much, yes neural networks (NNs) take longer and are more difficult to train, but they evaluate faster than regression trees and can take better advantage of GPUs for an even greater speed gain.

Furthermore, grid search, Bayesian hyperparameter tuning, genetic neural algorithms are all automatable hyperparameter tuning methods that would be simple to implement with your 1-D, low-dimensional system (though I am not suggesting you do that here).

~Line 85: "Since chemistry is inherently an embarassing parallel task, the speedup achieved on a single CPU as in this work will transfer - given large enough simulation grids making up for the overhead - on parallel computations."

Not sure what this means. Gas-phase chemistry is already parallelizable in air quality applications of atmospheric transport.

Paragraph starting at Line 210: "The choice of the regressor for each output is actually arbitrary, and nothing forbids to have different regressors for each variables, or even different regressors in different regions of parameter space of each variable. Without going into details on all kinds of algorithms that we tested, we found that decision-tree based methods such as Random Forest and their recent gradient boosting evolutions appear the most flexible and successful for our purposes. Their edge can in our opinion be resumed by: (1) implicit feature selection by construction, meaning that the algorithm automatically recognizes which input variables are most important for the estimation of the output; (2) no need for standardisation of both inputs and outputs; (3) ability to deal with any probability distributions; (4) fairly quick to train with sensible hyperparameter defaults; (5) extremely efficient implementations available."

I do not like the reasoning of this paragraph from an ML perspective. In terms of your reasoning:

1) The feature selection automation from RFs are not accurate if your inputs are colinear in any way, and I do not think your 7 features are completely independent of one another, 2) your next paragraph makes sense about this though you contradict it immediately in the following paragraph when writing about scaling outputs. I would not say that this is too much of an advantage, it just removes an annoyance, 3) most other ML algorithms other than Gaussian processes can deal with any type of distribution as they are (for the most part) nonparametric learners, 4) they are quicker to train than NNs but also slower to evaluate than a NN, 5) perhaps, but I would not say that a NN is much harder when using Python or R.

The main edge in terms of using an RF rather than a NN as a surrogate is that an RF will

always fall back on data it has seen and predict a mean state when it does not know what to output, whereas a NN is a regressor that will extrapolate outside of its training domain and accumulate errors much faster than an RF. Thus, RF (or xgboost) is typically more robust than a NN.

Figure 2, 12, 13: A little confused here, what are the axes labels?

~Line 278: Can xgboost use a GPU? I know that RFs cannot really use them but perhaps a factor of 10x gain may be had by using a GPU in the future.

Figure 14: Why do you not call the PHREEQC solver here? Did it never fail the mass balance criteria or is that not implemented here. Wondering what would happen if you routinely called the PHREEQC solver (e.g. at every 10% of simulation progress time), and ran it for several iterations if the physics solver can be used to dampen error?

General clarifications:

For your simulations, you call the full physics solver when the output variables fail the mass balance screening. Is it the case that once the surrogate fails the mass balance test the solver is just continually called or is there a dampening of error from using the full solver? Do you expect this relationship to change with a higher-dimensional system or higher grid resolution (2D, 3D model)?

I thought the mass balance screening + feature engineering justification + Figure 11 reasoning was excellent. Well done!