MS-No.: egusphere-2022-1362

Version: Revision

Title: Data-driven methods to estimate the committor function in conceptual ocean models

Author(s): Valérian Jacques-Dumas and co-authors

Point-by-point reply to reviewer #2

March 20, 2023

We thank the reviewer for their careful reading and for the useful comments and will adapt the manuscript accordingly. Below is a point-by-point reply with the referee's comments in bold font, our reply in italic font and the changes in manuscript in normal font.

1. **Figure 3: as described in the caption, there should be "confidence intervals" for FFNN training time in subplot (c). Or is it just too narrow to see?**

   The confidence intervals are plotted the same way on every subplot of Figure 3 and Figure 5. For FFNN in subplot (c), the confidence interval is plotted but indeed too narrow to see. The difference between the 5^{th} *and* 95^{th} *percentiles is less than* 8.5% *of the average training time for* $N_T \leq 30$ *and less than* 3.7% *of the average training time for* $N_T \geq 50$*. It can only be seen by zooming enough for* $N_T = 10$*.*

   We will add a sidenote in the caption of Figure 3 explaining that the confidence interval for FFNN is too narrow to see.

2. **It would be helpful to provide more details about the scaling issue in the AMG method. Overall AMG method seems to be rather competitive especially when the number of available data is limited (which is typically the case in real application). Does the computational bottleneck come from the increasingly expensive KD-tree search or the eigenvalue algorithms for the large analogue matrix? For the latter case, one can exploit the sparsity structure in the analogue matrix and use iterative eigen-solvers. Matrices of size 1e4 by 1e4 seem still efficient to work with.**

   In the case of AMC, the main computational bottleneck stems from the eigenvector algorithm. This step is already implemented using a SciPy sparse matrix function which is a wrapper to the ARPACK function. The latter is an efficient implementation of the Implicitly Restarted Arnoldi Method so AMC already uses an iterative solver and takes advantage of the sparsity of $\tilde{G}$*. This solver has, however, a complexity of* $O(N^2)$ *(if* N *is the number of samples in the training set) while building and searching through the Kd-tree does not have a larger complexity than* $O(N \log N)$*.*

   On page 17, we will provide additional details about the time complexity of AMC: "For AMC, building the Kd-tree is at worst $O(N \log N)$ while querying it is about $O(N)$ and the eigenvector search is $O(N^2)$. Thus when N grows large, the training time of AMC mainly depends on the eigenvector search in $\tilde{G}$, although we are using the Implicitly Restarted Arnoldi Method, which already takes advantage of the sparsity of $\tilde{G}$."

3. **The authors focus on the scaling issue from one perspective– the data dimension. However when applying the algorithm to other models, the variable dimension typically plays a critical role and efficient algorithms with moderate complexity growth with the model dimension is indeed the research target in the committor function community. I don't think it is necessary to provide numerical evidence in this angle but I would suggest the author include a brief discussion and clarify this different definition about "high-dimensional" problems.**

   We thank the reviewer for pointing this out, it is indeed an important distinction to make because the methods don't scale the same way whether we are looking at the data dimension or the variable dimension. For AMC, the dependence on the variable dimension only appears during the Kd-tree search but has a limited impact on the complexity. Concerning DGA, the impact of the variable dimension is also limited and only intervenes when computing the distance between the time steps. In the case of FFNN, however, the model dimension is directly related to the number of parameters to train and can have a significant impact on the performance of the network. For RC, the scaling with the model dimension is even worse since this dimension appears in the formula of the size of the reservoir, which is computed through a binomial product, so it may quickly blow-up.

   We will provide a more detailed discussion of the impact of the model dimension on each method in the "Discussion" section, on page 26.

4. **The discussion about computational time is rather long for all models. I would suggest adding theoretical scalings of, say $O(N)$, $O(N^2)$, etc. to figure 3 and 5 to make a clear visualization and comparison.**

   This is a very good suggestion. We call N the number of samples in the training set. Due to the eigenvector computation, AMC has a complexity $O(N^2)$. For DGA, it is also because of an eigenvector problem that the scaling is $(O(N^3))$. The complexity of both FFNN and RC only depends on their architecture and they are given the training samples sequentially, hence a complexity in $O(N)$. If we now call M the number of test samples, both FFNN and RC have a testing time scaling as $O(M)$ (they see the samples sequentially). Testing AMC mainly depends on building a Kd-tree from the training samples, hence its testing time scales at worst as $O(N \log N)$ and testing DGA requires extending the trained modes, which scales as $O(MN)$.

   Adding these scalings to figures 3 and 5 may overload them with information. Instead, we will provide details about these scalings in pages 17-18 at the beginning of the discussion about computational times.