

Dear Editor,

In this paper, the authors are explaining how a quite basic neural network (NN) can be used to speed up the generation of events following a given distribution. First, the authors used the neural network to approximate the function to integrate. Second, they generate un-weighted events according to the underlying distribution of the learned function. Third, they re-weight that sample to the original function. Finally, they un-weight their re-weighted sample. The gain in speed relies mainly on the assumption that the evaluation of the NN is much faster than the evaluation of the full matrix-element and it further assumes that if the NN approximates the true function correctly, the number of evaluations of the full matrix-element can be reduced significantly like in any importance sampling method.

The authors present two non-trivial examples of the above procedure for the generation of events at LHC using AMEGIC (part of the SHERPA package). The authors demonstrate a speed-up up to ten without any significant bias in the distribution. However, some processes are actually slower and the speed-up is also possible thanks to a more relaxed tolerance on the residual presence of over-weight event.

The paper is well written, it is interesting for the Pheno community, and it will likely create some follow-up paper. However, I do have a series of concerns which will require substantial modifications to the document. Consequently, I am recommending to ask the authors for a major revision of their work. When the authors will successfully answer those concerns and/or adapt their paper accordingly, I will be able to recommend the paper for publication.

Major points

1. As explained in the paper, their main algorithm (Algorithm number 2) is valid for any value of w_{max} . However, the name of that variable seems to suggest that it should be taken (ideally) as $max(w)$. Accordingly, the authors pick the value according to the distribution of w_i (Eq. 14 for example). However, such choices can lead to large over-weights, especially if $s_{max} \equiv max(s_i) > w_{max}$. Since the authors do not use the distribution of s_i to determine the value of w_{max} these over-weights can be quite large and are not under control. The authors entirely miss to comment on this source of over-weight and do not include it in their Kish Effective sample size. This should be corrected.
2. The usual implementation of multiple stage un-weighting is that the $(N + 1)$ un-weighting step tries to compensate for the over-weighted events generated by the N previous un-weighting steps. In other words, it is more common to find the following formula for a two stage un-weighting algorithm in the literature:¹

$$\tilde{w} = Max \left(X_{max}, \frac{w}{s} Max(s_{max}, s) \right) \quad (1)$$

with (ideally),

$$s_{max} \approx Max(s), \quad (2)$$

$$X_{max} \approx Max \left(\frac{w}{s} Max(s_{max}, s) \right). \quad (3)$$

In contrast, the formula used by the authors sounds less optimal (i.e. creating artificially more

¹Using here the fully normalized weight convention defined by the authors in eq.2 –see point 7–.

over-weight). The authors should explain why they do not use the standard formula and compare to it.

3. The authors are quite sparse on the interplay between their method and the multi-channel method. Maybe this is an issue that I do not know AMEGIC enough, but I fail to see how a single NN can be used in the presence of multi-channel. More details on that point will allow other groups to implement/test such method (this can be included in an appendix if this is too technical or too AMEGIC specific).
4. The physics validation is done on a sum of contributions (different initial/final state). However, from Table 4 and 6, it is clear that some of those contributions (the sub-leading ones) are associated to large over-weights (low α) and/or display a significant slow-down. The fact that they are sub-leading makes it difficult to assess if those contributions are also biased. In addition to their summed result, the authors should also produce differential cross-sections for the problematic contribution alone.

Minor points

5. The authors present their algorithm as "novel", but they fail to indicate that their method is nothing more than a quite standard importance sampling method. More comparisons on how their method extend the idea behind VEGAS algorithm would be useful for the large audience.
6. The conclusion should clearly indicate that their speed-up is associated to an over-weight that needs to be compensated by a larger sample size. Also, the fact that the speed-up is process dependent and that the procedure can fail for some processes, should be indicated in the conclusion.
7. The authors are using two different conventions for the weights of the events throughout the paper. First, they define it as a dimensionfull quantity, since their average equals the cross-section/integral (Eq. 2). But in most of the paper they use them as dimensionless quantity. The authors do not provide which value they used for this normalization, which is ambiguous in the presence of over-weights. While not that relevant in the paper, it is a crucial point for experimentalists to know how to normalize such type of samples.
8. The efficiency reported in the paper are precise and very objective. Unfortunately, they do not include the overhead needed by the method for generating the training sample and the actual training of the NN. A perfect metric would be to indicate what should be the minimal sample size needed to have a net gain, but simply indicating how long the additional steps related to the training take should be enough.

Additional suggestions to the authors

9. A more detailed discussion of other fitting methods (e.g. boosted decision trees) that the authors have tested would be interesting for the reader and would provide an initial answer to the question on how good the fitting procedure needs to be in order to provide an actual speed up (issue raised in the conclusion).
10. The authors could update their toy example and present a case where $w_{max} \neq s_{max}$.