How to re-interpret BDT-based LLP searches?

This is a live document, where people can go ahead and write out thoughts and ideas ahead of the discussion which will take place on Weds May 26th at ~15h CET: https://indico.cern.ch/event/980853/timetable/#b-420244-reinterpretation

Please feel free to contribute!

LC:

- Basic problem: searches which make use of machine-learning techniques are not generally easy to re-interpret in terms of models other than those in the original paper
    - In fact, (theorists please correct me if I'm wrong), I don't think ANY of the many LLP searches which have used BDTs/NNs in recent years have been re-used…
    - This is obviously a problem in the long-term: what happens in 40nok years when all the analysts have moved on? We will still want to exploit these searches !
    - Need to think now about this problem

- Should we be preserving BDTs/NNs on HEPData?
    - If so, can they easily be run ? (what happens in 40 years when no-one uses/maintains TMVA anymore??)
        - TMVA BDTs can quite easily be converted to pure C++/python code eg using https://gitlab.com/agbuckley/bdt2cpp/-/tree/master
        - This means no horrible dependencies need to be carried around just to re-run the BDTs
    - However, does similar system exist for NNs?
        - ONNX has been mentioned, but is it really as portable as we need it?

- Or do we need to preserve the analysis in a Docker container, with all dependencies?
    - This is what RECAST framework which is used by ATLAS does:
        - Does CMS have any similar containerisation projects?
        - And LHCb ?
    - How to give Theorists access to this ?

- Even if we *can* preserve BDTs, can they actually get re-run without a full detector simulation?
    - Eg if one uses detailed response eg # clusters, jet width etc, as inputs…
    - Could efficiency tables/smearing of truth variables do the jon?
        - Probably not, but are there other options?
    -
- How can we train our BDTs/NNs to avoid such issues?
    - Train on variables well-defined at truth-level, if possible
    - Smear inputs according to systematics to avoid overtraining to nominal..

- …?

Karri/Kate side conversation - which LL analyses actually use event level BDTs/ML?

Most common seems to be object level

- CMS displaced jet NN tagger
  http://cms-results.web.cern.ch/cms-results/public-results/publications/EXO-19-011/
- CMS displaced jets
  http://cms-results.web.cern.ch/cms-results/public-results/publications/EXO-19-021/

Could we not just provide parameterized efficiencies here? This is what we do for b-tagging. For reinterpretation efficiencies don't need to be perfect anyway, just "good enough"

Any event level?

- Maybe ATLAS calRatio?? LC : yes, we do !
- https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PAPERS/EXOT-2017-25/
- https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-PHYS-PUB-2020-007/

Hot take: we shouldn't use ML for LL searches :)
👀

From Zoom Chat

Giordon S: SModelS also works with the full likelihoods -- not just simplified ones.
Jan Heisig: Exactly, haven't have time to comment on it. So far there are three ATLAS analyses where we use this information.
Andy Buckley: Credit where it's due (rather than break the flow of discussion): Louie himself has added some very nice features to that bdt2cpp tool!
G S: This paper from Ghosh, Nachman ,Whiteson -- https://arxiv.org/abs/2105.08742 -- has some times on uncertainty-aware ML.
G S: ONNX: https://onnx.ai/
G S: You can see the supported tools: https://onnx.ai/supported-tools.html
G S : @Andy:
https://github.com/microsoft/onnxruntime/blob/master/docs/Versioning.md#:~:text=Compatibility-,Backwards%20compatibility,range%20%5B7%2D9%5D.
Andy Buckley: The NN preserving tool that Harrison P mentioned wasn't in the Reint Forum, actually, but on a mailing list. He was talking about Frugally Deep:
https://github.com/Dobiasd/frugally-deep

Andy Buckley: I think there are more, e.g. this https://github.com/serizba/cppflow (and IIRC PyTorch also has a C/C++ interface)