Template fits for semileptonic analysis in LHCb

Julián García Pardiñas[1]

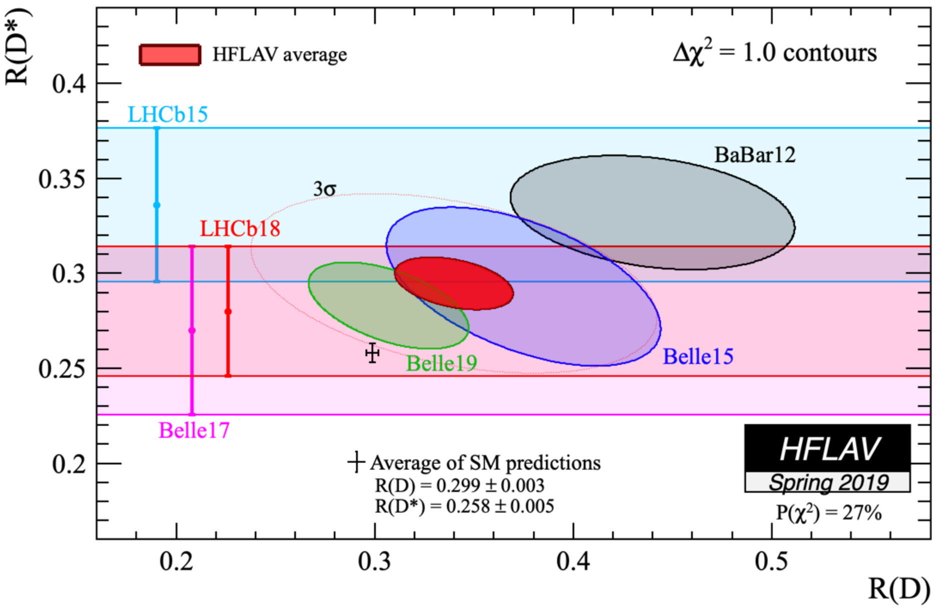
[1]Universität Zürich (UZH), Switzerland

Binned and template fits in TensorFlow
CERN - 25/06/2019

Aim of this talk

LHCb is using template fits to **study semileptonic b-hadron decays**, such as B→D*lν.

- Very large data samples.
- Templates from simulation and control data.
- **Some shape parameters not well known, we want to float them during fitring**.



- ☆ The common approach in LHCb is to rely on **HistFactory**.
- ☆ To solve some limitations, I've been working on implementing an interface between HistFactory and a newly developed package, **HAMMER**.
- ☆ **Idea: using HAMMER (or a similar implementation) in TensorFlow?**
  - Different scenarios can be considered.
  - I will try to asses some key points for this goal from my (limited) experience.

Parametric shapes vs. templates

To perform binned fits with a variable-shape model, one can essentially follow two possible paths.

Parametric shapes

- Use a **phenomenological expression** for the decay rate, as a function of some “true” variables.
- Account for **efficiency/resolution via some kind of response matrix**.

- Pros: fast to evaluate.
- Cons: need for a reliable response matrix; difficult to transform the expressions to use some more elaborated variables.

Templates

- Take the shapes from **simulation or control data samples**.
- To change the shape, **re-weight the events** (new model/old model) and re-evaluate the histograms.

- Pros: efficiency and resolution automatically accounted for, any variable can be used.
- Cons: the reweighing would be slow for large samples.

I will focus on the **templates**, trying to overcome their limitations.

Recap. on HistFactory

Interesting features

- **Easy to construct** rather complicated combinations of species/channels.

- Propagation of **template-statistics uncertainties** through the HistFactory implementation of the Barlow-Beeston method.

- Some flexibility for the template shapes by introducing interpolation between modified histograms.

Limitations

- The **histogram interpolation is not powerful enough** in general to accurately determine the parameters of a relatively complicated model.
→ Improvement with HAMMER, next slide.

- The ROOT implementation of HistFactory is a bit obscure and there is **very limited control on the likelihood**.
→ Improvement with pyhf? (I don't have experience with it).

The HAMMER package

Helicity Amplitude Module
for Matrix Element Reweighting

http://hammer.physics.lbl.gov

The HAMMER tool provides variable-shape templates with fast evaluation (for example, for fitting).

It is still not fully finished (official release not available yet).

Two-step procedure (done internally by HAMMER):

- **“Slow” step:** pre-processing of a MC sample, examining the histogram variation as a function of the desired parameters and parameterizing it via analytical expansions.

- **“Fast” step:** computation the template for some given values of the parameters, via the previously-obtained expansions.

caching exercise

HAMMER is internally **coded in C++** but also has **python wrappers** for interfacing. It needs to know the **new/old phenomenological model** for the species (so far, only some models have been coded).

Combination with HistFactory: I have created some new interfacing classes to use HAMMER inside HistFactory, allowing full shape variation for the desired templates. Still not fully finished, some checks ongoing.

Ideas for a TensorFlow implementation

At Zürich, Abhijit Mathad and Martina Ferrillo have started looking into possibilities for how to best profit from the previous functionality in TensorFlow.

In general terms, if HAMMER is used, it should be interfaced in two moments:
- Before the fit, to pre-process the samples.
- During fitting, at each minimization step (and before the likelihood evaluation), to retrieve the cached elements and construct the updated model.

I have quickly asked one of the HAMMER developers about their thoughts on expanding to TensorFlow and he has told me that it would be easy to wrap the python wrapper into TensorFlow. So some further iteration in this sense can be interesting.

If HAMMER is not used, some TensorFlow-based similar caching could be a possibility.

Conclusions

- A priori, variable-shape data-driven templates require slow per-event re-weighting.
- This can be avoided with some pre-computating and caching.
- HAMMER does this job efficiently via cached analytical expansions.
- This would be a very interesting functionality to have available in TensorFlow.