Gammapy modeling

Christoph Deil (MPIK)
Jan 5, 2017

Status

- **gammapy.utils.modeling** - our own Parameter and ParameterList class; some experimental code for XML serialisation

- **gammapy.spectrum.models** - some models using gammapy.utils.modeling, party with to_sherpa converter

- **gammapy.image.models** - uses **Astropy models**; not used much, 2D image modeling tutorials use **Sherpa models** directly

- **gammapy.cube** has some Sherpa models (used so far) and started CombinedModel3D (no parameter handling yet)

- In short: a big mess

Options

- There are different good options, especially **Astropy** and **Sherpa** provide modeling / fitting frameworks. Both have a powerful modeling language to create complex and composite models, and many pre-implemented models.

- It is possible to extend Astropy & Sherpa and add what we need, but it’s not very nice. For the built-in models an additional problem is where to pass x/y or lon/lat.

- There are other options that should be mentioned, but that I won’t discuss today: astromodels, saba, Fermi ST, Gammalib, tensorflow …

Short-term Plan

- Change existing code to use gammapy.utils.modeling
  Milestone: no more import gammapy.modeling and import sherpa

- Add XML & YAML serialisation
  Milestone: read CTA 1DC and gamma-cat models as Gammapy objects

- Add simple model evaluation on maps
  Milestone: a flux and predicted counts cube of the CTA 1DC GPS survey

- For fitting, have a simple, common interface to optimisers.
  Milestone: a fit with scipy.optimize and/or iminuit and/or Sherpa and/or emcee

- Write down a PIG and start work this week.
  The work can be split among a few people. Interested?

Long-term plan

- The short-term plan will allow most studies / use cases for H.E.S.S. and CTA

- But it will not be efficient computationally efficient.
  There's a ton of tricks one can do, e.g. use gradients, avoid evaluating the same multiple times (i.e. cache parts of model evaluation), analytical special cases, …

- In the future, there could be whole new modeling frameworks, either in Gammapy or on top of Gammapy.
  E.g. implement models in TensorFlow and use a CPU/GPU cluster to achieve orders of magnitude speedup over what Gammapy, but also others like Fermi ST or ctools do.

- And we're also not going to have a nice modeling language that supports linked parameters (hard to add, very hard to serialise) and model composition via expressions (easy to add).

- I think we can just re-evaluate feature / performance / user feedback / developer manpower and if we want build something better, probably while keeping the end-user modeling interface, i.e. without annoying users.