A **Python** package for **gamma-ray** astronomy

ERLANGEN CENTRE FOR ASTROPARTICLE PHYSICS

Prospects for an unbinned analysis framework in Gammapy

Tim Unbehaun, Gammapy Coding Sprint – Jan 2022

Binned vs Unbinned

A Python package for gamma-ray astronomy

ERLANGEN CENTRE FOR ASTROPARTICLE PHYSICS

- Information loss in binned analyses
  → faster computation for large event numbers
  → less sensitive in low statistics case
- Keep as much information as possible from event reconstruction to final fit
  → Gain ~20% more sensitivity

Binned likelihood

$$\ln \mathcal{L}(\xi) = \sum_{i=1}^{N} \ln \left[\underbrace{\frac{\nu_i(\xi)^{n_i}}{n_i!} \times \exp(-\nu_i(\xi))}_{\text{Poisson probability to find n counts when } \nu \text{ are predicted}} \right]$$

sum over pixels

Unbinned likelihood

$$\ln \mathcal{L}(\xi) = \sum_{e=1}^{N} \ln \left[\underbrace{\frac{dN_i(\xi)}{d\Omega \times dE_{reco}}}_{\text{Probability of event to belong to given model}} \right] - N^{tot}(\xi)$$

total number of predicted counts

sum over events

Unbinned analysis as done in Fermi

copying from “Unbinned Maximum Likelihood for LAT Data” by J. Chiang (GSFC-UMBC)

integrals over solid angle and energy
prob of energy dispersion
prob of PSF
effective area
predicted model flux

The likelihood model, M, describes the expected distribution of photons,

$$M(E', \hat{p}', t) = \boxed{\int dE\, d\hat{p}}\, \boxed{D(E'; E, \hat{p}, \vec{L}(t))}\, \boxed{P(\hat{p}'; E, \hat{p}, \vec{L}(t))}\, \boxed{A(E, \hat{p}, \vec{L}(t))}\, \boxed{S(E, \hat{p})} \quad (4)$$

$$\equiv \int dE\, d\hat{p}\, R(E', \hat{p}', t; E, \hat{p}) S(E, \hat{p}). \quad (5)$$

The latter relation defines the function R, referred to hereafter as the “total response”. The source model consists of point sources and diffuse emission,

$$S(E, \hat{p}) = \sum_i s_i(E) \delta(\hat{p} - \hat{p}_i) + S_G(E, \hat{p}) + S_{\rm eg}(E, \hat{p}). \quad (6)$$

The index i labels the individual point sources; $s_i(E)$ is the true energy spectrum of source i; and $\hat{p}_i$ is its location on the sky. S_G is the Galactic diffuse component, and $S_{\rm eg}$ is the extragalactic diffuse component. Note that the $s_i(E)$ have dimensions of $dN/dEdtdA$ while S_G and $S_{\rm eg}$ have dimensions of $dN/dEdtdAd\Omega$.

Labeling individual photon events with the index j, the logarithm of the Poisson likelihood is

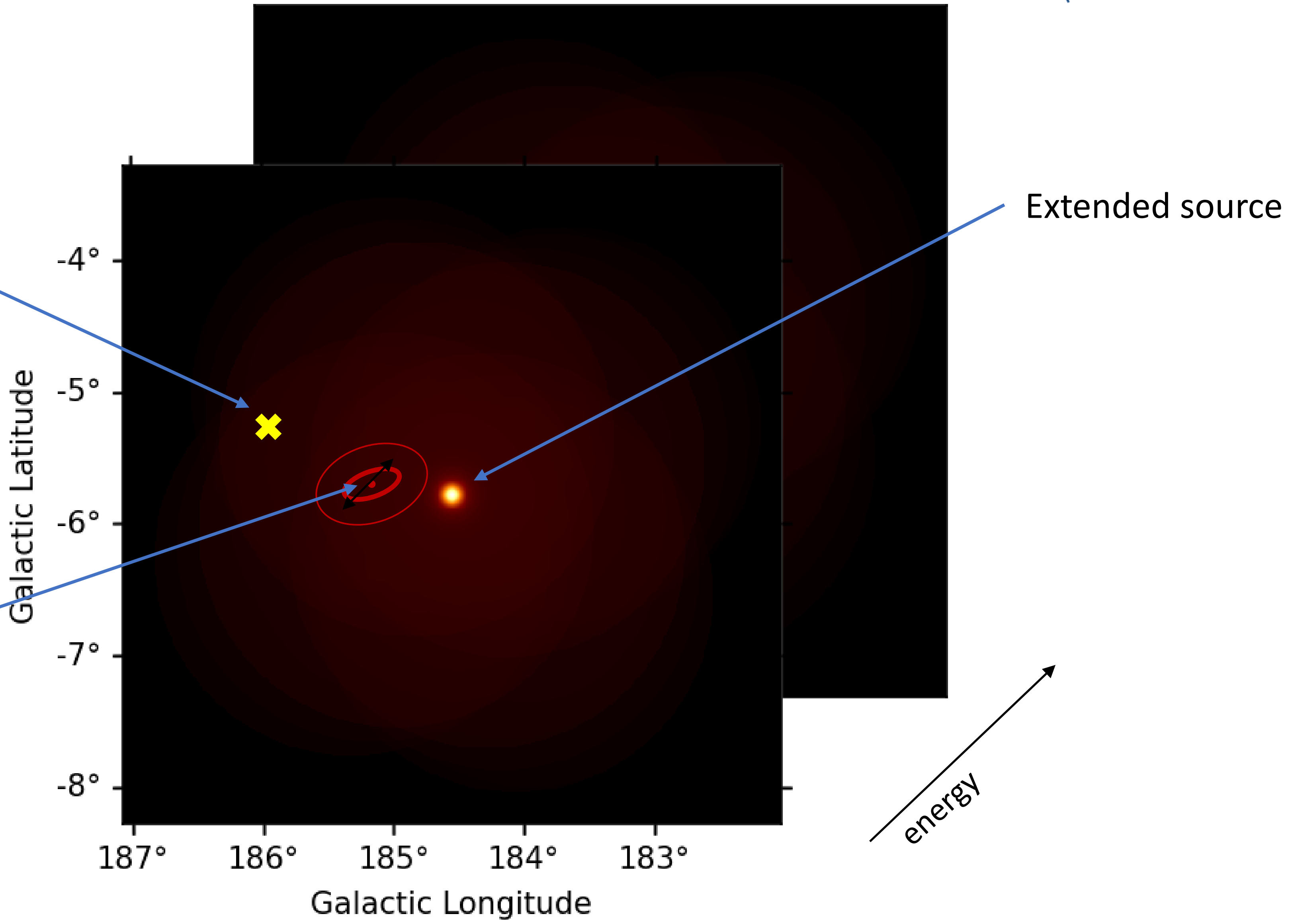
$$\log \mathcal{L} = \sum_j \log M(E'_j, \hat{p}'_j, t_j) - N_{\rm pred}, \quad (7)$$

$$N_{\rm pred} = \int dE'\, d\hat{p}'\, dt\, M(E', \hat{p}', t).$$

Unbinned analysis as done in Fermi

Point source

Event with uncertainties in energy and space
Uncertainties from IRFs (averaged uncertainties) or directly from event reconstruction (loose less information)



Unbinned analysis as done in Fermi

copying from “Unbinned Maximum Likelihood for LAT Data” by J. Chiang (GSFC-UMBC)

Splitting of the integral: Calculation of total response R on the fly (size constraints)
→ only re-calculate the parts where parameters changed, cache the rest

integral for point sources (no spatial int needed)

integral for extended sources and background

point source integral for N^{tot}

extended source/bkg integral for N^{tot}

In particular, if we define

$$a_{ij} \equiv \int dE\, s_i(E)\, R(E'_j, \hat{p}'_j, t_j; E, \hat{p}_i) \quad (9)$$

$$b_j \equiv \int dE\, d\hat{p}\, [S_G(E,\hat{p}) + S_{\rm eg}(E,\hat{p})]\, R(E'_j, \hat{p}'_j, t_j; E, \hat{p}) \quad (10)$$

$$c_i \equiv \int dE\, s_i(E) \int dE'\, d\hat{p}'\, dt\, R(E', \hat{p}', t; E, \hat{p}_i) \quad (11)$$

$$d \equiv \int dE\, d\hat{p}\, [S_G(E,\hat{p}) + S_{\rm eg}(E,\hat{p})] \int dE'\, d\hat{p}'\, dt\, R(E', \hat{p}', t; E, \hat{p}), \quad (12)$$

the log-likelihood is

$$\log \mathcal{L} = \sum_j \log\left(\sum_i a_{ij} + b_j\right) - \sum_i c_i - d. \quad (13)$$

Thoughts on implementation in GP

New data set class like "EventDataset":

- Event properties as table (array for calculations)
- PDFs from IRFs / event uncertainties (need fast evaluations)
- Maybe integration limits per event based on PDFs (like 99% containment)
- Models (with mask of events where sep > $r_{99}^{\text{model}} + r_{99}^{\text{event}}$)

Existing framework,
maybe improve speed

Because of size limitations we will probably need to loop through all events
→ use numba? + good speed, easy to do parallel for-loop
- the whole framework would need to be wrapped into numba (interpolation of IRFs, evaluation of models, etc…)

→ Use caching for the contributions of each model component and only compute relevant events

→ How to do fast but accurate integrations?

Thoughts on implementation in GP

New Event-Evaluator class or set of functions:

- Will be called from the stat_sum function for each model component
- Receives:
  - Contributing Events as array
  - IRF values and bins as arrays ready for interpolation
  - Evaluated Model on fine geometry (with axes) as array for interpolation
- Loop through events, interpolate IRFs and model, integrate PDFs
- Could be wrapped in numba framework
- Maybe do the caching from the Dataset? Could probably also done by Evaluator class…

- How to deal with unit handling?

Thoughts on different strategies

strategy 1)
- fine binned predicted counts cube
- integration over true geom is forward folding with IRFs
- interpolation of npred at E'_j, p'_j
  → npred for each event

The likelihood model, M, describes the expected distribution of photons,

$$M(E',\hat{p}',t) \;=\; \int dE\, d\hat{p}\, D(E';E,\hat{p},\vec{L}(t))P(\hat{p}';E,\hat{p},\vec{L}(t))A(E,\hat{p},\vec{L}(t))S(E,\hat{p})$$

$$\log\mathcal{L} = \sum_j \log M(E'_j,\hat{p}'_j,t_j) - N_{\mathrm{pred}},$$

strategy 2)
- M(E’,p’) is probability cube
- D(E’),P(p’) are inverse IRFs for backfoling onto model geom
- integration over PDFs * model flux
- likelihood for each event

+ easy to implement, can use large amount of existing framework
- should converge (no improvement) to very fine “binned likelihood”
- where is the Poisson statistic?; Fit will maximize the npred at events positions while keeping $npred^{tot}$ low – Is the “meaning” of TS value still valid? (Ratio tests, etc…)
- npred for each event should be proportional to its probability

- difficult to implement, can’t use much of existing suff
  - Inverse IRFs are not available, atleast offset dependence is in true offset and one might need reco offset, rest should be ok since alpha is symmetric and edisp can be renormed
- more intuitive, really using probabilities
  - Maybe its just the more complicated backwards folding and result is same as strategy 1