ModelManipulFit

November 18, 2019

1 How to do advanced model manipulation and fitting in Python

In this tutorial you will learn to perform some more advanced model manipulation and fitting from Python. You can use these methods in an interactive Python session to explore your dataset and improve the known model of the sky. You can also use them in a Python pipeline to test multiple model hypotheses and select the one that is most appropriate.

As usual start by importing the gammalib, ctools, and cscripts Python modules.

```
In [ ]: import gammalib
        import ctools
        import cscripts
```

You may want to use the matplotlib package for plotting.

```
In [ ]: %matplotlib inline
        import matplotlib.pyplot as plt
```

1.1 Simulated dataset

For the tutorial you will simulate a small dataset. Start by defining the Instrument Response Function and energy range that will be used all along.

```
In [ ]: caldb = 'prod3b-v2'
        irf   = 'South_z40_0.5h'
        emin  = 0.1   # TeV
        emax  = 160.0 # TeV
```

Now proceed to simulate the dataset. It consists of an hour of observations of the Crab nebula region, as usual pointed at a slightly offset position from the target. The input model is different from the one used in the demonstration and contains some surprises. Don't look at it until you have completed the exercises at the end of the tutorial.

```
In [ ]: evfile = 'events_advanced.fits'

        obssim = ctools.ctobssim()
        obssim['ra']        = 83.63
        obssim['dec']       = 22.51
        obssim['rad']       = 5.0
        obssim['tmin']      = 0
        obssim['tmax']      = 3600
        obssim['emin']      = emin
        obssim['emax']      = emax
        obssim['caldb']     = caldb
        obssim['irf']       = irf
        obssim['inmodel']   = '$CTOOLS/share/models/crab_beyond.xml'
        obssim['outevents'] = evfile
        obssim.execute()
```

Peek at the simulated events by creating a skymap.

```
In [ ]: skymap = ctools.ctskymap()
```

1.2 Model fitting and residual inspection

Perform an unbinned fit to the data using the simple Crab model from the demonstration we did before.

```
In [ ]: like = ctools.ctlike()
        like['inobs']   = evfile
        like['caldb']   = caldb
        like['irf']     = irf
        like['inmodel'] = '$CTOOLS/share/models/crab.xml'
        like.run()
```

Let's look at the output from the optimizer. In this case it's useful to store the best-fit minus log-likelihood value for later usage.

```
In [ ]: print(like.opt())
        like1 = like.opt().value()
```

Check the fitted model. Is the background model in use consistent with the data?
Generate a residual map. Tip: look at the residuals in units of significance.

1.3 Adding model components

From the residual map there should be an obvious excess displaced w.r.t. the main emission peak. Is this another source?

To test this hypothesis we add a point source at the position we eye-ball from the residual map. We can do this by creating a new instance of a gammalib sky model component.

```
In [ ]: newpntsrc = gammalib.GModelSky(gammalib.GModelSpatialPointSource(83.7,21.9),
                                       gammalib.GModelSpectralPlaw(1.e-17,-2.,gammalib.GEnergy
                                       gammalib.GModelTemporalConst(1))
```

You have defined a sky model object that has three components:

- a spatial model, which is a point source at the position guessed from the residual map;

- a spectral model which is a power law with spectral index 2, and a flux which approximately 1/10 of the Crab nebula;
- a temporal model which is a constant.

Name this source Src1, and free its position (fixed by default when a new source is created), so that you can fit it to the data:

```
In [ ]: newpntsrc.name('Src1')
        newpntsrc['RA'].free()
        newpntsrc['DEC'].free()
```

Finally append the new source to the model container.

```
In [ ]: like.obs().models().append(newpntsrc)
```

and fit the model including the new source to the data.

```
In [ ]: like.run()
```

Does the addition of the new source provide a better fit to the data? You can quantify this using the test statistic (TS) given by twice the log-likelihood difference.

```
In [ ]: like2 = like.opt().value()
        ts    = -2.0 * (like2 - like1)
        print(ts)
```

TS is expected to be distributed as a chi-squared with n degrees of freedom, where n is the additional number of degrees of freedom in the model including the new source, in our case 4 (RA, Dec, Prefactor, and Index). The integral of the chi-squared from TS to infinity is the chance probability that the likelihood improved by that much due to statistical fluctuations. A large value, like the one we got, means that the chance probability is very low, thus we are likely to have found a new source.

To make sure the new source improves the data/model agreement look again at the residual map.

The addition of the new point source has flattened the spatial residuals, even though the fit is obviously not perfect yet. From now on fix the position of Src1.

```
In [ ]: like.obs().models()['Src1']['RA'].fix()
        like.obs().models()['Src1']['DEC'].fix()
```

1.4 Modifying model components

Next we want to check the residual spectrum, to see if our model spectrum satisfactorily reproduces the data. Tip: look at the region around the Crab with a radius of about 0.2 degrees.

From the residuals it should be clear that the model does not reproduce the data well. There is an excess at low energies and the model overshoots the data at high energies.

You may try to change the spectral model for the Crab nebula from a simple power law to a power law with exponential cutoff.

```
In [ ]: crab    = like.obs().models()['Crab']
        expplaw = gammalib.GModelSpectralExpPlaw()
        expplaw['Prefactor'].value(crab['Prefactor'].value())
        expplaw['Index'].value(crab['Index'].value())
        expplaw['PivotEnergy'].value(crab['PivotEnergy'].value())
        expplaw['CutoffEnergy'].value(1.e6)
        crab.spectral(expplaw)
```

Run a new fit and make sure the new spectrum improves the results. Tip: look both at the likelihood value and at the spectral residuals.

Although there was an improvement, now you should see that seemingly there is a missing component at high energies. You may try to add a new power-law spectral component for the Crab on top of the exponentially-cutoff power law.

```
In [ ]: expplaw2  = like.obs().models()['Crab'].spectral().clone()
        newcomp   = gammalib.GModelSpectralPlaw(1.e-18,-2.,gammalib.GEnergy(1,'TeV'))
        comp_spec = gammalib.GModelSpectralComposite()
        comp_spec.append(expplaw2)
        comp_spec.append(newcomp)
        like.obs().models()['Crab'].spectral(comp_spec)
```

Again, run a fit and make sure the newly-added component makes the model a better description of the data. Tip: it should be the case. However, is this the best possible model for our region?

1.5 For further excercise

- Test other spectral models for the Crab nebula, such as log-parabola and broken power law. Can you say what the best spectral model is?
- What can you say from the spectral residual spectra about the spectrum of Src1?
- Replace the 2 sources with a single extended source (disk). Can you say which of the two hypotheses is describing the data better?