Astropy_fitting

October 20, 2016

1 Astropy models and fitting

If you need to do least square fitting for data to a model a good place to start is `astropy`'s modeling and fitting code.

1.1 Packages being used

- `astropy`: for modeling and fitting
- `matplotlib`: for plotting

1.2 Relevant documentation

- `astropy`: http://docs.astropy.org/en/stable/modeling/index.html

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        from astropy.modeling import models, fitting
        import mpl_style
        %matplotlib inline
        plt.style.use(mpl_style.style1)
```

1.3 1-D model fitting

For an example lets look at the problem of fitting a 1-D model to a spectral line. First we need to create some fake data:

```
In [2]: x = np.linspace(-5., 5., 200)
        y = 3 * np.exp(-0.5 * (x - 1.3)**2 / 0.8**2)
        y += np.random.normal(0., 0.2, x.shape)
```

1.3.1 A trapezoid model

```
In [3]: t_init = models.Trapezoid1D(amplitude=1.0, x_0=0.0, slope=0.5)
        fit_t = fitting.LevMarLSQFitter()
        t = fit_t(t_init, x, y)
        print(t)
```

```
Model: Trapezoid1D
Inputs: ('x',)
Outputs: ('y',)
Model set size: 1
Parameters:
      amplitude        x_0          width         slope
    ------------- ------------- ------------- -------------
    2.96403789662 1.31554465735 0.32841114655 1.80237844144
```

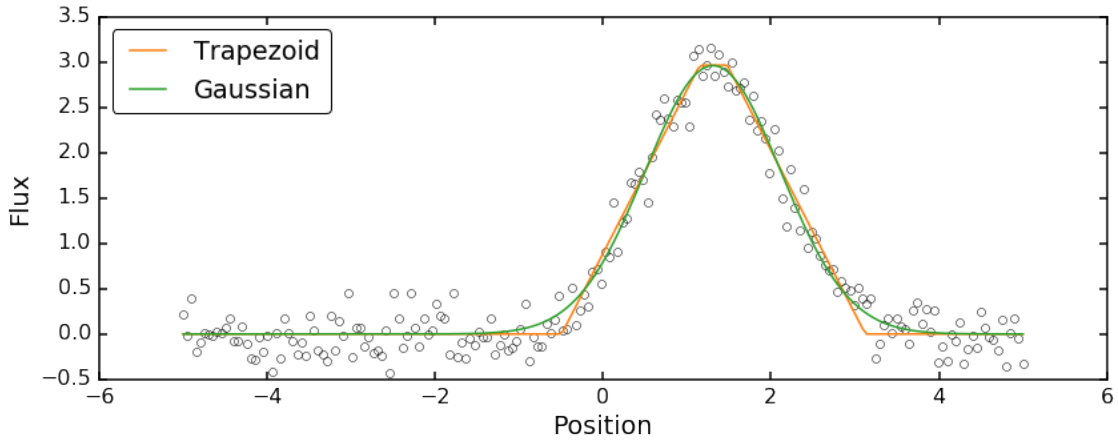
1.3.2 A Gaussian model

```
In [4]: g_init = models.Gaussian1D(amplitude=1., mean=0, stddev=1.)
        fit_g = fitting.LevMarLSQFitter()
        g = fit_g(g_init, x, y)
        print(g)
```

```
Model: Gaussian1D
Inputs: ('x',)
Outputs: ('y',)
Model set size: 1
Parameters:
      amplitude         mean          stddev
    ------------- ------------- --------------
    2.95986731307 1.30957309516 0.807327726979
```

1.3.3 Plotting the results

```
In [5]: plt.figure(1, figsize=(12,5))
        plt.plot(x, y, 'o', mfc='none')
        plt.plot(x, t(x), label='Trapezoid')
        plt.plot(x, g(x), label='Gaussian')
        plt.xlabel('Position')
        plt.ylabel('Flux')
        plt.legend(loc=2)
        plt.tight_layout()
```



1.4 Compound models

Models can also be 'added' together before fitting. To demonstrate lets make a new dataset made up to two Gaussians.

```
In [6]: np.random.seed(42)
        g1 = models.Gaussian1D(1, 0, 0.2)
        g2 = models.Gaussian1D(2.5, 0.5, 0.1)
        x = np.linspace(-1, 1, 200)
        y = g1(x) + g2(x) + np.random.normal(0., 0.2, x.shape)
```

1.4.1 Make the model

The model can be 'added' just like arrays:

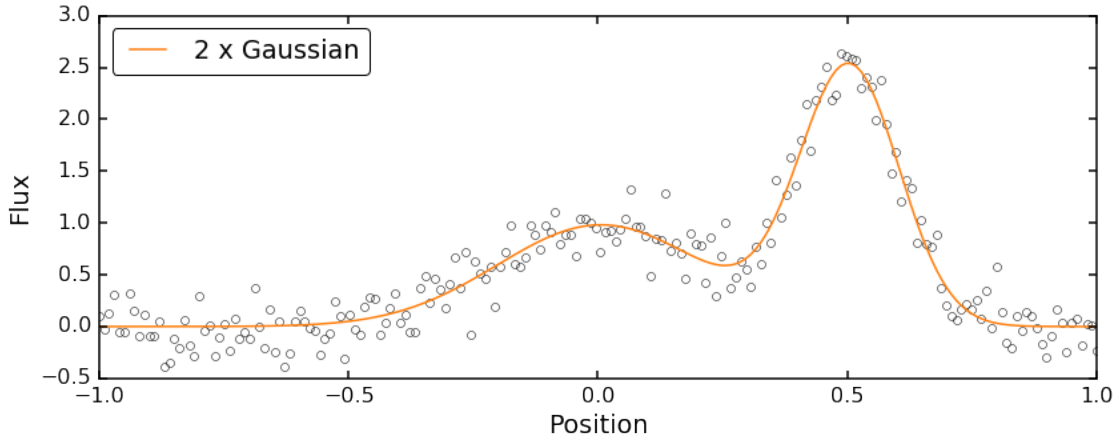
```
In [7]: gg_init = models.Gaussian1D(1, 0, 0.1) + models.Gaussian1D(2, 0.5, 0.1)
        fit_gg = fitting.SLSQPLSQFitter()
        gg = fit_gg(gg_init, x, y)
        print(gg)

Optimization terminated successfully.    (Exit mode 0)
            Current function value: 6.83285936044
            Iterations: 14
            Function evaluations: 137
            Gradient evaluations: 14
Model: CompoundModel0
Inputs: ('x',)
Outputs: ('y',)
Model set size: 1
Expression: [0] + [1]
Components:
    [0]: <Gaussian1D(amplitude=1.0, mean=0.0, stddev=0.1)>

    [1]: <Gaussian1D(amplitude=2.0, mean=0.5, stddev=0.1)>
Parameters:
     amplitude_0        mean_0       ...     mean_1         stddev_1
    -------------- ---------------- ... -------------- ---------------
    0.981184943263 0.00597741767885 ... 0.504121855043 0.0999995837067
```

1.4.2 Plot the result

```
In [8]: plt.figure(2, figsize=(12, 5))
        plt.plot(x, y, 'o', mfc='none')
        plt.plot(x, gg(x), label='2 x Gaussian')
        plt.xlabel('Position')
        plt.ylabel('Flux')
        plt.legend(loc=2)
        plt.tight_layout()
```



1.5 `Astropy`'s models

`Astropy` has a large number of 1- and 2-D models built in. Check out http://docs.astropy.org/en/stable/modeling/index.html#module-astropy.modeling.functional_models for a full list. If the model you are looking for is not built in, you can always define your own: http://docs.astropy.org/en/stable/modeling/new.html.

1.6 Limitations

- Uses OLS (or similar) to maximize an objective function (and all the assumptions about the data that go into this, e.g. Gaussian errors)
- Cov of fit only returned for some fitters (fond on the `fitter.fit_info()` method)

In [ ]: