Forward Modeling Code

Craig Lage

November 2, 2015

1 Basic Idea

We start with a number N of pixelized images, each $N_x x N_y$ pixels in size. these images are referred to as "postage stamps". We want to simultaneously fit all of these images with a single function, typically a 2D Gaussian. We assume that this function has the same shape for all of the postage stamps, but we allow the magnitude of the function to vary from stamp to stamp, and we also allow an offset of the origin of the fitting function which varies from stamp to stamp. Let M_{nij} be the measured data of each postage stamp, where n is an index that identifies the stamp, and i and j are indices which identify the X-Y location within the stamp. Let f_{nij} be the normalized fitting function. The form of the fitting function does not matter for this analysis, but we next illustrate a typical fitting function. This fitting function for a 2D Gaussian is of the following form, with p the pixel size, and $xoff_n$ and $yoff_n$ the offsets of the Gaussian function origin from the pixel center.

$$f_{nij} = I = \frac{1}{4}(\mathrm{erf}(\frac{xh}{\sqrt{2}\sigma_x}) - \mathrm{erf}(\frac{xl}{\sqrt{2}\sigma_x}) * (\mathrm{erf}(\frac{yh}{\sqrt{2}\sigma_y}) - \mathrm{erf}(\frac{yl}{\sqrt{2}\sigma_y})) \quad (1)$$

where:

$$xh = (i + \frac{1}{2})p + xoff_n; xl = (i - \frac{1}{2})p + xoff_n; yh = (j + \frac{1}{2})p + yoff_n; yl = (j - \frac{1}{2})p + yoff_n \quad (2)$$

Since we allow each stamp to have a different intensity, the function that we want to minimize to find the best fit is as follows:

$$F = \sum_n^N \sum_i^{N_x} \sum_j^{N_y} (M_{nij} - I_n f_{nij})^2 \quad (3)$$

Expanding:

$$F = \sum_n^N \sum_i^{N_x} \sum_j^{N_y} (M_{nij}^2 - 2M_{nij} I_n f_{nij} + I_n^2 f_{nij}^2) \quad (4)$$

Since we are minimizing this function, we can take the partial derivatives with respect to the I_n and set them equal to zero. This gives N equations, each of which can be solved for one of the coefficients I_n, giving:

$$I_n = \frac{\sum_i^{N_x} \sum_j^{N_y} M_{nij} f_{nij}}{\sum_i^{N_x} \sum_j^{N_y} f_{nij}^2} \quad (5)$$

In the code, I use the following shorthand notations (the subscript n is implied and not used in the code):

$$a2_n = \sum_i^{N_x} \sum_j^{N_y} M_{nij}^2 \quad (6)$$

$$\mathrm{b2_n} = \sum_{\mathrm{i}}^{\mathrm{N_x}} \sum_{\mathrm{j}}^{\mathrm{N_y}} \mathrm{f^2_{nij}} \tag{7}$$

$$\mathrm{ab_n} = \sum_{\mathrm{i}}^{\mathrm{N_x}} \sum_{\mathrm{j}}^{\mathrm{N_y}} \mathrm{M_{nij} f_{nij}} \tag{8}$$

So we can then write:

$$\mathrm{I_n} = \frac{\mathrm{ab_n}}{\mathrm{a2_n}} \tag{9}$$

We can now substitute the result for $\mathrm{I_n}$ back in to the expression for F, giving:

$$\mathrm{F} = \sum_{\mathrm{n}}^{\mathrm{N}} (\sum_{\mathrm{i}}^{\mathrm{N_x}} \sum_{\mathrm{j}}^{\mathrm{N_y}} \mathrm{M^2_{nij}} - 2 \frac{\sum_{\mathrm{i}}^{\mathrm{N_x}} \sum_{\mathrm{j}}^{\mathrm{N_y}} \mathrm{M_{nij} f_{nij}}}{\sum_{\mathrm{i}}^{\mathrm{N_x}} \sum_{\mathrm{j}}^{\mathrm{N_y}} \mathrm{f^2_{nij}}} \sum_{\mathrm{i}}^{\mathrm{N_x}} \sum_{\mathrm{j}}^{\mathrm{N_y}} \mathrm{M_{nij} f_{nij}} + (\frac{\sum_{\mathrm{i}}^{\mathrm{N_x}} \sum_{\mathrm{j}}^{\mathrm{N_y}} \mathrm{M_{nij} f_{nij}}}{\sum_{\mathrm{i}}^{\mathrm{N_x}} \sum_{\mathrm{j}}^{\mathrm{N_y}} \mathrm{f^2_{nij}}})^2 \sum_{\mathrm{i}}^{\mathrm{N_x}} \sum_{\mathrm{j}}^{\mathrm{N_y}} \mathrm{f^2_{nij}}) \tag{10}$$

Which, using the shorthand notation, can be expressed as:

$$\mathrm{F} = \sum_{\mathrm{n}}^{\mathrm{N}} (\mathrm{a2_n} - 2 \frac{\mathrm{ab_n}}{\mathrm{b2_n}} \mathrm{ab_n} + (\frac{\mathrm{ab_n}}{\mathrm{b2_n}})^2 \mathrm{b2_n}) = \sum_{\mathrm{n}}^{\mathrm{N}} (\mathrm{a2_n} - \frac{\mathrm{ab^2_n}}{\mathrm{b2_n}}) \tag{11}$$

This is the form that is implemented in the forward.cpp python extension. It takes as inputs a list of N postage stamps and values of σ_x and σ_y, and returns the intensity for each stamp $\mathrm{I_n}$, as well as the result F, which is the sum-of-squares fit between the measured postage stamps and the fitting function. The value of F is then used in a steepest descent algorithm (Powell's method) to find the values of σ_x and σ_y which give the best fit to the measured data.