Fitting an Ellipse to a Set of Data Points

Nicky van Foreest

1 THE PROBLEM

Given a set of points $\mathbf{x}_i = (x_i, y_i)$ find the best (in a least squares sense) ellipse that fits the points.

Other interesting pages that discuss this topic:

- `http://bec.physics.monash.edu.au/wiki/Main/FittingEllipses`
- `http://exnumerus.blogspot.com/2011/03/python-script-to-fit-ellipse-to-noisy.html`

Note, the code below is much shorter than the code discussed on this last page, but perhaps less generic. For instance, it does not seem to work well for circles.

Even though I never use this code myself, it turns out to be quite a 'big' hit. I wrote it in 2006 to test a numerical integrator for the following case. Suppose we are flying in a rocket in a circular orbit around the earth. If we hit the break for a very short time the orbit must become elliptical: we lose kinetic energy, but as the potential energy remains the same, the rocket must reach the same height after completing an orbit. Thus, to check python's numerical integrator, I used the integrator to compute a rocket in an elliptical orbit. Thus I used the code to check that the orbit was indeed an ellipse.

The code is in the file `fit_ellipse.py`.

2 THE APPROACH

We follow an approach suggested by Fitzgibbon, Pilu and Fischer in Fitzgibbon, A.W., Pilu, M., and Fischer R.B., *Direct least squares fitting of ellipsees*, Proc. of the 13th Internation Conference on Pattern Recognition, pp 253--257, Vienna, 1996. I learned of this approach from Peter Snoeren, whose development I present below.

An ellipse can be defined as a general conic, that is, as the set of points $\mathbf{x} = (x, y)$ such that

$$f(a, (x, y)) = D \cdot a = 0 \tag{1}$$

where $D = (x^2, xy, y^2, x, y, 1)$ and $a = (a_{x,x}, a_{x,y}, a_{y,y}, a_x, a_y, a_1)$[1].

We can fit the ellipse to N data points $\mathbf{x}_i$, $i = 1, \ldots, N$, by minimizing the distance

$$\Delta(a, \mathbf{x}) = \sum_{i=1}^{N} (f(a, \mathbf{x}_i))^2.$$

Rewriting this we obtain:

$$\Delta(a, \mathbf{x}) = \sum_{i=1}^{N} a^T D_i^T D_i a = a^T S a$$

[1] Compare `http://mathworld.wolfram.com/QuadraticCurve.html`

where $S_i = \sum D_i^T D_i$ is a 6×6 scatter matrix.

The aim is find the a that minimizes $\Delta(a, \mathbf{x})$ but such that an ellipse results. Now it is known that the components of the vector a have to satisfy the constraint

$$4a_{x,x}a_{y,y} - a_{x,y}^2 > 0$$

if the corresponding conic is to be an ellipse. Written as a matrix equation and using the above vectorial form for a this condition can be rewritten to $a^T C a > 0$,

$$C = \begin{pmatrix} 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Next, observe that the conic condition $f(a, \mathbf{x}) = 0$, eq. (1), is independent of linear scaling in a. We therefore may just as well replace this condition by $a^T C a = \phi$, where ϕ is some positive number.

Combining the above results in the constrained minimization problem

$$\text{argmin}_a \{\Delta(a, \mathbf{x}) : a^T C a = \phi\}$$

In other words, the vector a that solves this minimization problem corresponds to the ellipse we are looking for.

The standard method to solve the above constrained problem is to introduce a Lagrange multiplier λ and a Lagrangian

$$L(a, \lambda) = \Delta(a, \mathbf{x}) - \lambda(a^T C a - \phi) = a^T S a - \lambda(a^T C a - \phi).$$

and minimize $L(a, \lambda)$ as a function of a.

To solve this, take gradient with respect to a and the derivative with respect to λ:

$$\begin{aligned} \partial_a L(a, \lambda) = 0 &\implies 2Sa - \lambda Ca = 0 \implies Sa = \lambda Ca \\ \partial_\lambda L(a, \lambda) = 0 &\implies a^T C a = \phi. \end{aligned}$$

Multiplying the first equaton from the left with a^T and using the second equation we obtain that

$$a^T S a = \lambda a^T C a = \lambda \phi$$

Cleary, ϕ is arbitrary, but constant. Thus, to minimize $\Delta(a, \mathbf{x}) = a^T S a$ we are looking for the smallest λ that satisfies this equation.

Finally, observe that $Sa = \lambda Ca$ can be rewritten as a generalized eigenvalue problem

$$\frac{1}{\lambda} a = S^{-1} C a.$$

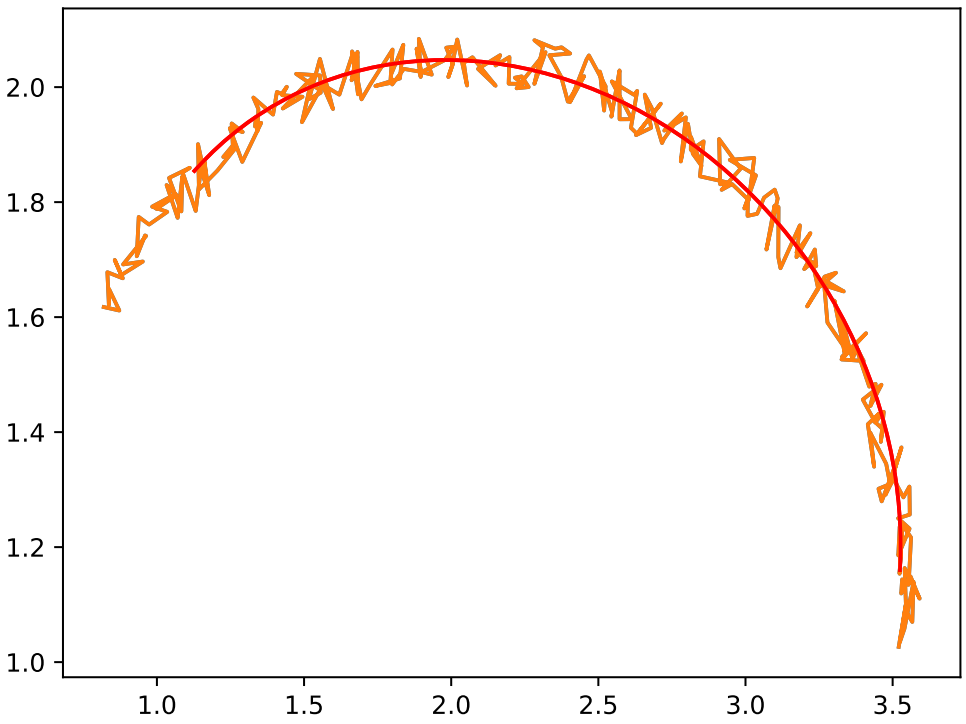
Thus, we have to solve this eigenvalue problem, and the a we are looking for is the eigenvector corresponding to the largest eigenvalue $1/\lambda$.

Now that we have the minimizing a we can use a to compute the center of the ellipse, its angle of rotation and its axes. The relations between a and these characteristics can be found at `http://mathworld.wolfram.com/Ellipse.html`.

3 AN EXAMPLE

Finally consider the following example:

```
from fit_ellipse import *

arc = 0.8
R = np.arange(0,arc*np.pi, 0.01)
x = 1.5*np.cos(R) + 2 + 0.1*np.random.rand(len(R))
y = np.sin(R) + 1. + 0.1*np.random.rand(len(R))

a = fitEllipse(x,y)
center = ellipse_center(a)
phi = ellipse_angle_of_rotation(a)
axes = ellipse_axis_length(a)

a, b = axes
xx = center[0] + a*np.cos(R)*np.cos(phi) - b*np.sin(R)*np.sin(phi)
yy = center[1] + a*np.cos(R)*np.sin(phi) + b*np.sin(R)*np.cos(phi)

import matplotlib.pylab as plt

plt.plot(x,y)
plt.plot(xx,yy, color = 'red')
```



Not too bad altogether. By increasing the arc the results improve (that is, some simple testing shows this).