Direct linear least squares fitting of an ellipse

Posted by: christian (/blog/author/christian/) on 9 Aug 2021

(15 comments)

Fitting a set of data points in the xy plane to an ellipse is a suprisingly common problem in image recognition and analysis. In principle, the problem is one that is open to a linear least squares solution, since the general equation of any conic section can be written

$$F(x, y) = ax^2 + bxy + cy^2 + dx + ey + f = 0,$$

which is linear in its parameters a, b, c, d, e and f. The polynomial $F(x, y)$ is called the *algebraic distance* of any point (x, y) from the conic (and is zero if (x, y) happens to lie *on* the conic).

A naive algorithm might attempt to minimize the sum of the squares of the algebraic distances,

$$d(\boldsymbol{a}) = \sum_{i=1}^{N} F(\boldsymbol{x}_i)^2,$$

with respect to the parameters $\boldsymbol{a} = [a, b, c, d, e, f]^{\mathrm{T}}$. However, there are two problems to overcome: firstly, such an algorithm will almost certainly find the trivial solution in which all parameters are zero. To avoid this, the parameters must be constrained in some way. A common choice, noting that $F(x, y)$ can be multiplied by any non-zero constant and still represent the same conic, is to insist that $f = 1$ (other algorithms set $a + c = 1$ or $||\boldsymbol{a}||^2 = 1$).

The second problem however, is that there is no guarantee that the conic best fitted to an arbitrary set of points is an ellipse (for example, their mean squared algebraic distance to a hyperbola might be smaller). This is more likely to be the case where the points only lie near elliptical arc rather than around an entire ellipse. The appropriate constraint on the parameters of $F(x, y)$ in order for the conic it represents to be an ellipse, is $b^2 - 4ac < 0$. Some iterative algorithms seek to minimise $F(x, y)$ in steps, ensuring this constraint is met before each step.

However, a direct least squares fitting to an ellipse (using the algebraic distance metric) was demonstrated by Fitzgibbon et al. (1999) (https://www.researchgate.net/publication/2885417_Direct_Least_Square_Fitting_of_Ellipses). They used the fact that the parameter vector $\boldsymbol{a}$ can be scaled arbitrarily to impose the *equality* constraint $4ac - b^2 = 1$, thus ensuring that $F(x, y)$ is an ellipse. The least-squares fitting problem can then be expressed as minimizing $||\boldsymbol{D}\boldsymbol{a}||^2$ subject to the constraint $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{a} = 1$, where the *design matrix*

$$\boldsymbol{D} = \begin{pmatrix} x_1^2 & x_1 y_1 & y_1^2 & x_1 & y_1 & 1 \\ x_2^2 & x_2 y_2 & y_2^2 & x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^2 & x_n y_n & y_n^2 & x_n & y_n & 1 \end{pmatrix}$$

represents the minimization of F and the *constraint matrix*

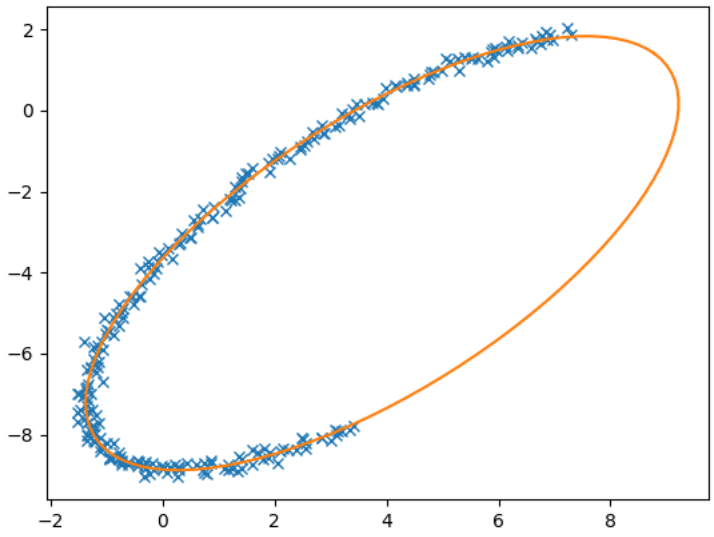
$$\boldsymbol{C} = \begin{pmatrix} 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

expresses the constaint $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{a} = 1$. The method of Lagrange multipliers (as introduced by Gander (1980) (https://link.springer.com/article/10.1007/BF01396656)) yields the conditions:

$$\begin{aligned} \boldsymbol{S}\boldsymbol{a} &= \lambda \boldsymbol{C}\boldsymbol{a} \\ \boldsymbol{a}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{a} &= 1, \end{aligned}$$

where the *scatter matrix*, $\boldsymbol{S} = \boldsymbol{D}^{\mathrm{T}}\boldsymbol{D}$. There are up to six real solutions, $(\lambda_j, \boldsymbol{a}_j)$ and, it was claimed, the one with the smallest positive eigenvalue, λ_k and its corresponding eigenvector, $\boldsymbol{a}_k$, represent the best fit ellipse in the least squares sense.

Fitzgibbon et al. provided an algorithm, in MATLAB, implementing this approach, which was subsequently improved by Halíř and Flusser (1998) (http://autotrace.sourceforge.net/WSCG98.pdf) for reliability and numerical stability. It is this improved algorithm that is provided in Python below.



Note that the algorithm is inherently biased towards smaller ellipses because of its use of the algebraic distance as a metric for the goodness-of-fit rather than the geometric distance. As noted by both sources for the algorithm used in this post, this issue is discussed by Kanatani (1994) (https://ieeexplore.ieee.org/document/276132) and is not easily overcome.

```
import numpy as np
import matplotlib.pyplot as plt


def fit_ellipse(x, y):
    """

    Fit the coefficients a,b,c,d,e,f, representing an ellipse described by
    the formula F(x,y) = ax^2 + bxy + cy^2 + dx + ey + f = 0 to the provided
    arrays of data points x=[x1, x2, ..., xn] and y=[y1, y2, ..., yn].

    Based on the algorithm of Halir and Flusser, "Numerically stable direct
    least squares fitting of ellipses'.


    """

    D1 = np.vstack([x**2, x*y, y**2]).T
    D2 = np.vstack([x, y, np.ones(len(x))]).T
    S1 = D1.T @ D1
    S2 = D1.T @ D2
    S3 = D2.T @ D2
    T = -np.linalg.inv(S3) @ S2.T
    M = S1 + S2 @ T
    C = np.array(((0, 0, 2), (0, -1, 0), (2, 0, 0)), dtype=float)
    M = np.linalg.inv(C) @ M
    eigval, eigvec = np.linalg.eig(M)
    con = 4 * eigvec[0]* eigvec[2] - eigvec[1]**2
    ak = eigvec[:, np.nonzero(con > 0)[0]]
    return np.concatenate((ak, T @ ak)).ravel()


def cart_to_pol(coeffs):
    """

    Convert the cartesian conic coefficients, (a, b, c, d, e, f), to the
    ellipse parameters, where F(x, y) = ax^2 + bxy + cy^2 + dx + ey + f = 0.
    The returned parameters are x0, y0, ap, bp, e, phi, where (x0, y0) is the
    ellipse centre; (ap, bp) are the semi-major and semi-minor axes,
    respectively; e is the eccentricity; and phi is the rotation of the semi-
    major axis from the x-axis.

    """

    # We use the formulas from https://mathworld.wolfram.com/Ellipse.html
    # which assumes a cartesian form ax^2 + 2bxy + cy^2 + 2dx + 2fy + g = 0.
    # Therefore, rename and scale b, d and f appropriately.
    a = coeffs[0]
    b = coeffs[1] / 2
    c = coeffs[2]
    d = coeffs[3] / 2
    f = coeffs[4] / 2
    g = coeffs[5]

    den = b**2 - a*c
    if den > 0:
        raise ValueError('coeffs do not represent an ellipse: b^2 - 4ac must'
                         ' be negative!')

    # The location of the ellipse centre.
    x0, y0 = (c*d - b*f) / den, (a*f - b*d) / den

    num = 2 * (a*f**2 + c*d**2 + g*b**2 - 2*b*d*f - a*c*g)
    fac = np.sqrt((a - c)**2 + 4*b**2)
    # The semi-major and semi-minor axis lengths (these are not sorted).
    ap = np.sqrt(num / den / (fac - a - c))
    bp = np.sqrt(num / den / (-fac - a - c))

    # Sort the semi-major and semi-minor axis lengths but keep track of
    # the original relative magnitudes of width and height.
    width_gt_height = True
    if ap < bp:
        width_gt_height = False
        ap, bp = bp, ap

    # The eccentricity.
    r = (bp/ap)**2
    if r > 1:
```

```
        r = 1/r
    e = np.sqrt(1 - r)

    # The angle of anticlockwise rotation of the major-axis from x-axis.
    if b == 0:
        phi = 0 if a < c else np.pi/2
    else:
        phi = np.arctan((2.*b) / (a - c)) / 2
        if a > c:
            phi += np.pi/2
    if not width_gt_height:
        # Ensure that phi is the angle to rotate to the semi-major axis.
        phi += np.pi/2
    phi = phi % np.pi

    return x0, y0, ap, bp, e, phi


def get_ellipse_pts(params, npts=100, tmin=0, tmax=2*np.pi):
    """
    Return npts points on the ellipse described by the params = x0, y0, ap,
    bp, e, phi for values of the parametric variable t between tmin and tmax.

    """

    x0, y0, ap, bp, e, phi = params
    # A grid of the parametric variable, t.
    t = np.linspace(tmin, tmax, npts)
    x = x0 + ap * np.cos(t) * np.cos(phi) - bp * np.sin(t) * np.sin(phi)
    y = y0 + ap * np.cos(t) * np.sin(phi) + bp * np.sin(t) * np.cos(phi)
    return x, y


if __name__ == '__main__':
    # Test the algorithm with an example elliptical arc.
    npts = 250
    tmin, tmax = np.pi/6, 4 * np.pi/3
    x0, y0 = 4, -3.5
    ap, bp = 7, 3
    phi = np.pi / 4
    # Get some points on the ellipse (no need to specify the eccentricity).
    x, y = get_ellipse_pts((x0, y0, ap, bp, None, phi), npts, tmin, tmax)
    noise = 0.1
    x += noise * np.random.normal(size=npts)
    y += noise * np.random.normal(size=npts)

    coeffs = fit_ellipse(x, y)
    print('Exact parameters:')
    print('x0, y0, ap, bp, phi =', x0, y0, ap, bp, phi)
    print('Fitted parameters:')
    print('a, b, c, d, e, f =', coeffs)
    x0, y0, ap, bp, e, phi = cart_to_pol(coeffs)
    print('x0, y0, ap, bp, e, phi = ', x0, y0, ap, bp, e, phi)

    plt.plot(x, y, 'x')     # given points
    x, y = get_ellipse_pts((x0, y0, ap, bp, e, phi))
    plt.plot(x, y)
    plt.show()
```

Current rating: 4.9 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 Rate

← Chaotic Balls (/blog/chaotic-balls/) The Dottie number → (/blog/the-dottie-number/)

Comments

Comments are pre-moderated. Please be patient and your comment will appear soon.

Rafal 2 years, 1 month ago

Thanks a lot for the python implementation. It works perfect and is very handy.

Link (/blog/direct-linear-least-squares-fitting-of-an-ellipse/#comment-543) | Reply

Current rating: 5 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 Rate

DK 1 year, 5 months ago

Here is one solution if one has to discretize the ellipse for given x or given y values:

```
def discretize_ellipse(coeffs, x=None, y=None):
"""Get points from ellipse by given x/y. Applies the ellipse formular
ax^2 + bxy + cy^2 + dx + ey + f = 0 rearranged for x/y

Args:
coeffs (list): ellipse parameters in cartesian format
x (float, np.ndarray, list, optional): x-values to get the y's for.
y (float, np.ndarray, list, optional): y-values to get the x's for.

Returns:
tuple(np.ndarray, np.ndarray): The discrete points for given coordinate.
It will be x1, x2 if y was passed or y1,y2 if x was passed.
"""
a, b, c, d, e, f = coeffs
if x is None:
# formular rearranged for x
y = np.array([y]).flatten()
t1 = (-b*y - d)
t2 = np.sqrt( (b*y + d)**2 - 4*a*(c*y**2 + f + e*y) )
t3 = (2*a)
return ( t1 + t2 ) / t3, ( t1 - t2 ) / t3
elif y is None:
# formular rearranged for y
x = np.array([x]).flatten()
t1 = (-b*x - e)
t2 = np.sqrt((b*x + e)**2 - 4*c*(a*x**2 + d*x + f))
t3 = (2*c)
return (t1 + t2) / t3, (t1 - t2) / t3
```

Link (/blog/direct-linear-least-squares-fitting-of-an-ellipse/#comment-636) | Reply

Currently unrated ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 Rate

Hennadii 1 year, 5 months ago

The article mentions the method of Lagrange multipliers introduced by Gande (1981). Can someone provide a reference, I couldn't find the publicaiton it with Google.

Link (/blog/direct-linear-least-squares-fitting-of-an-ellipse/#comment-645) | Reply

Currently unrated ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 Rate

christian 1 year, 5 months ago

Thanks for getting in touch– I think the reference should have been Gander (1980): https://link.springer.com/article/10.1007/BF01396656 (https://link.springer.com/article/10.1007/BF01396656) – Least squares with a quadratic constraint, Numerische Mathematik 36, 291 (1980).

I've fixed this in the article and added a link.

Link (/blog/direct-linear-least-squares-fitting-of-an-ellipse/#comment-646) | Reply

Currently unrated ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 Rate

Doug 1 year, 2 months ago

Nice and nice references.

Link (/blog/direct-linear-least-squares-fitting-of-an-ellipse/#comment-667) | Reply

Currently unrated ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 Rate

Armadillo 1 year, 1 month ago

At end of 2nd paragraph, "A common choice...is to insist that f=1 (other algorithms set a+c=1 or ||a||^2=1)."

What is the justification for setting a+c =1?

Can you provide any references that use these other algorithms (that set a+c=1 or ||a||^2=1)?

Link (/blog/direct-linear-least-squares-fitting-of-an-ellipse/#comment-692) | Reply

Currently unrated ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 Rate

christian 1 year, 1 month ago

Hello,
You can look at the cited articles of the ones mentioned in the article for more information: for example, P. L. Rosin, "A note on the least squares fitting of ellipses", Pattern Recognition Letters 14, 799 (1993) discusses the relative merits of choosing f=1 and a+c=1
(https://www.sciencedirect.com/science/article/abs/pii/0167865593900621
(https://www.sciencedirect.com/science/article/abs/pii/016786559390062I))
Cheers, Christian

Link (/blog/direct-linear-least-squares-fitting-of-an-ellipse/#comment-694) | Reply

Currently unrated ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 Rate

Chris Pook 10 months, 4 weeks ago

Hello, thanks for posting this code. I found it while trying to resolve an issue I was having with the skimage EllipseModel() function, which also uses the algorithm from Halíř and Flusser (1998). I have thousands of ellipses from microscopy and found that the skimage function failsed to fit an elliptical model to maybe a quarter of them for no obvious reason.
See StackOverflow & GitHub posts:
https://stackoverflow.com/questions/75227988/why-can-i-model-some-of-these-ellipses-with-skimage-and-not-others (https://stackoverflow.com/questions/75227988/why-can-i-model-some-of-these-ellipses-with-skimage-and-not-others)
https://github.com/scikit-image/scikit-image/issues/6699 (https://github.com/scikit-image/scikit-image/issues/6699)

You can download a Jupyter Notebook to reproduce this here:
https://gist.github.com/chrispook/3fb15068dc7dcfb0a4ec6e3c4e768852
(https://gist.github.com/chrispook/3fb15068dc7dcfb0a4ec6e3c4e768852)

I have encountered similar issues with ellipse fitting to my data using the code on this page. See this Jupyter Notebook: https://gist.github.com/chrispook/56b7ed348301939489bb8ccae84c2736
(https://gist.github.com/chrispook/56b7ed348301939489bb8ccae84c2736)

I think it is highly unlikely that the developers of skimage and yourself have both introduced a bug into your code. Instead I suspect there may be a fundamental flaw in the algorithm of Halíř and Flusser (1998). I am not a mathematician though so cannot state that with any certainty.

Regards,
Chris

Link (/blog/direct-linear-least-squares-fitting-of-an-ellipse/#comment-719) | Reply

Currently unrated ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 Rate

christian 10 months, 4 weeks ago

Hi Chris,
I have replied by email to you about this: the explanation I think is that your data points consist of valu
differing by small relative amounts from their means which leads to numerical errors in the matrix inv
due to the finite precision of floating point numbers: you can subtract off the mean (shifting the origin
your points) to make the algorithm more reliable.
Cheers, Christian

Link (/blog/direct-linear-least-squares-fitting-of-an-ellipse/#comment-720) | Reply

Current rating: 5 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 Rate

Ahram 10 months, 1 week ago

Hi Christian, thanks for nice posting!
I think this algorithm is variant for translation and scaling, though. If I l translate or scale the input data, it gives different ellipse and far worse, it fails some cases. I am still looking for affine transformation-invariant ellipse fitting algorithm.

Link (/blog/direct-linear-least-squares-fitting-of-an-ellipse/#comment-722) | Reply

Currently unrated ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 Rate

christian 10 months, 1 week ago

Hello Ahram,
It's a little hard to know what to say about this without some examples. If it isn't an issue with the numerical stability of the algorithm implementation (one can always scale or transform the data into some place where fitting will fail), it would be interesting to see the specific issue you are experiencing.
Cheers, Christian

Link (/blog/direct-linear-least-squares-fitting-of-an-ellipse/#comment-723) | Reply

Currently unrated ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 Rate

Angel 7 months, 1 week ago

Hello, thank you for the information. Does this equation give any sense as to the errors in result? Or how would you go about seeing how well this ellipse fits besides just looking at the ellipse it generated?

Link (/blog/direct-linear-least-squares-fitting-of-an-ellipse/#comment-746) | Reply

Currently unrated ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 Rate

christian 7 months, 1 week ago

There is no unique choice for this: the presented algorithm minimizes the "algebraic distance" of the fitted ellipse from the data points and this could be used as a measure of the goodness-of-fit (i.e. "errors"). An alternative would be to use the "geometric distance". The article contains a reference (Kanatani (1994)) which discusses the difference.

Link (/blog/direct-linear-least-squares-fitting-of-an-ellipse/#comment-747) | Reply

Currently unrated ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 Rate

Ulysses 6 months, 4 weeks ago

Good stuff, Christian. You got me out of a bind and I appreciate it. A few comments:

* `inv(a) @ b` is better written as `solve(a, b)`.
* `con = 4 * eigvec[0]*` could use a space.
* The ValueError talks about 4ac, but `den` uses 1ac. Since it's a denominator, may as well require >= 0.
* The Python docs recommend math.fmod() for floats.

Link (/blog/direct-linear-least-squares-fitting-of-an-ellipse/#comment-748) | Reply

Current rating: 5 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 Rate

christian 6 months, 4 weeks ago

Thanks for these suggestions: I've a feeling that the code from this post could more usefully be on GitHub and packaged, so I'll see if I can find time to do that.

Link (/blog/direct-linear-least-squares-fitting-of-an-ellipse/#comment-749) | Reply

Currently unrated ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 Rate

New Comment

Name

required

Email

required (not published)

Website

optional

Comment

required

Comment