A Simple Algorithm for Fitting a Gaussian Function

Article *in* IEEE Signal Processing Magazine · September 2011

DOI: 10.1002/9781118316948.ch31

1 author:

Hongwei Guo
Shanghai University
75 PUBLICATIONS **1,862** CITATIONS

SEE PROFILE

Hongwei Guo

A Simple Algorithm for Fitting a Gaussian Function

"DSP Tips and Tricks" introduces practical design and implementation signal processing algorithms that you may wish to incorporate into your designs. We welcome readers to submit their contributions. Contact Associate Editors Rick Lyons (R.Lyons@ieee.org) or Clay Turner (clay@claysturner.com).

Gaussian functions are suitable for describing many processes in mathematics, science, and engineering, making them very useful in the fields of signal and image processing. For example, the random noise in a signal, induced by complicated physical factors, can be simply modeled with the Gaussian distribution according to the central limit theorem from the probability theory. Another typical example in image processing is the Airy disk resulting from the diffraction of a limited circular aperture as the point-spread function of an imaging system. Usually an Airy disk is approximately represented by a two-dimensional Gaussian function. As such, fitting Gaussian functions to experimental data is very important in many signal processing disciplines.

This article proposes a simple and improved algorithm for estimating the parameters of a Gaussian function fitted to observed data points.

GAUSSIAN CURVE FITTING

Recall that a Gaussian function is of the form

$$y = Ae^{-(x-\mu)^2/2\sigma^2}. \tag{1}$$

This function can be graphed with a symmetrical bell-shaped curve centered at the position $x = \mu$, with A being the height of the peak and σ controlling its width, and on both sides of the peak the tails (low-amplitude portions) of the curve quickly fall off and approach the x-axis. The focus of this article is on how we fit a Gaussian function to observed data points and determine the parameters, A, μ, and σ exactly. The centroid method takes advantage of the symmetry of a Gaussian function, thus allowing determining the Gaussian peak position very efficiently [1], [2]. Although this method is popularly used in image processing for the subpixel peak detection of a point or a line, it does not enable us to estimate the width or height of a peak. In practice, it is not easy to determine all the Gaussian parameters including A, μ, and σ because this problem is generally associated with the solution of an overdetermined system of nonlinear equations, which is generated by substituting the observed data into (1).

The standard solution for such a nonlinear problem is to employ an iterative procedure like Newton-Raphson algorithm, with which a sufficiently good initial guess is crucial for correctly solving for the unknowns, and it is possible that the procedure does not converge to the true solution [3]. By noting that a Gaussian function is the exponential of a quadratic function, a simpler method was proposed by Caruana et al. [4]. It calculates the natural logarithm of the data first and then fits the results to a parabola. Another method is to fit the straight line resulting from the differential of the quadratic function just mentioned [5], [6]. With these algorithms, however, noise in the observed data may induce relatively large errors in the estimated parameters, and the accuracies strongly depend on the y-amplitude range of the observed data points.

To overcome the aforementioned problems, this article analyzes the effects of noise on Caruana's algorithm and derives a simple and improved technique for estimating the Gaussian parameters. The technique uses a weighted least-squares method, deduced from a noise model, to fit the logarithm of Gaussian data. As such, the influences of the statistical fluctuations on the estimated Gaussian parameters are significantly reduced. Based on this principle, we also suggest an iterative procedure that is considerably less sensitive to the amplitude range of the observed data points. As a result, the initial guesses for the estimated parameters will no longer be critical. We proceed by reviewing Caruana's algorithm in the next section.

CARUANA'S ALGORITHM

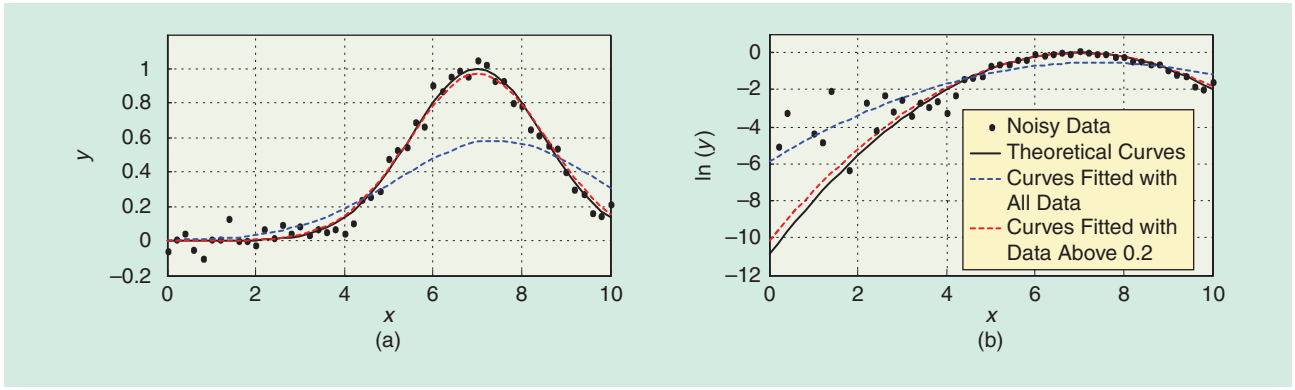
Caruana's algorithm is based on the fact that a Gaussian function is the exponential of a quadratic function. Taking the natural logarithm of the Gaussian function in (1) yields

$$\begin{aligned}\ln(y) &= \ln(A) + \frac{-(x-\mu)^2}{2\sigma^2} \\ &= \ln(A) - \frac{\mu^2}{2\sigma^2} + \frac{2\mu x}{2\sigma^2} - \frac{x^2}{2\sigma^2} \\ &= a + bx + cx^2, \end{aligned} \tag{2}$$

where $a = \ln(A) - \mu^2/(2\sigma^2)$, $b = \mu/\sigma^2$, and $c = -1/(2\sigma^2)$. By doing this, the nonlinear equation with unknowns A, μ, and σ is transformed into a linear one with unknowns being a, b, and c, thus alleviating its computational complexity. The Gaussian parameters A, μ, and σ can be calculated from a, b, and c.

Note that (2) denotes a parabola whose peak position is the same as that of the Gaussian function described in (1). We show an example of this in Figure 1, where the black solid curve in

Digital Object Identifier 10.1109/MSP.2011.941846
Date of publication: 22 August 2011



[FIG1] Parts (a) and (b) show the results of Caruana's algorithm in the presence of noise.

Figure 1(a) illustrates a Gaussian function with $A = 1, \mu = 7,$ and $\sigma = 1.5$, and the black solid curve in Figure 1(b) plots its logarithm.

The fundamental principle of Caruana's algorithm is to fit this parabola in Figure 1(b) in the least squares sense so that its coefficients, *a, b,* and *c* are determined, and then the Gaussian parameters, A, μ, and σ, are calculated as we shall show. To perform this task, an error function based on (2) is defined, namely

$$\delta = \ln(y) - (a + bx + cx^2). \tag{3}$$

Differentiating the sum of δ^2 with respect to a, b, and c and setting the resultant expressions to zero yields a linear system of equations

$$\begin{bmatrix} N & \sum x & \sum x^2 \\ \sum x & \sum x^2 & \sum x^3 \\ \sum x^2 & \sum x^3 & \sum x^4 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sum \ln(y) \\ \sum x \ln(y) \\ \sum x^2 \ln(y) \end{bmatrix}, \tag{4}$$

where N is the number of observed data points and $\sum$ denotes $\sum_{n=1}^{N}$ for shortening the expression. After solving (4) for a, b, and c, the desired parameters of the Gaussian function are calculated using

$$\mu = \frac{-b}{2c}, \tag{5}$$

$$\sigma = \sqrt{\frac{-1}{2c}}, \tag{6}$$

and

$$A = e^{a - b^2/4c}. \tag{7}$$

EFFECTS OF NOISE

Caruana's algorithm is computationally efficient, since it is noniterative. In the presence of noise, however, its accuracy decreases dramatically. See Figure 1 for an example, where the dots in Figure 1(a) denote the data obtained by sampling the black solid curve. The observed data is contaminated by zero-mean random noise having a standard deviation (SD) of 0.05. Excluding the negative-valued data points, their logarithms are plotted in Figure 1(b), also with dots. We see from them that the fluctuations of the data from their theoretical values, induced by the noise, may be magnified by the logarithmic operation, especially for the points far away from μ having small Gaussian values falling down. Using Caruana's algorithm, we fit the logarithmic data to a quadratic function. The resulting parabola is plotted in Figure 1(b) with the blue dashed curve, which noticeably deviates from the theoretical curve. The estimates of the Gaussian parameters are $A = 0.5946$, $\mu = 7.6933$, and $\sigma = 2.3768$, and the reconstructed Gaussian curve is shown in Figure 1(a), also with the dashed blue curve. From these results, the errors induced by noise are apparent. This phenomenon can be theoretically explained by considering an additive noise model.

If there is an additive random noise η, the data we observed is not the ideal value y but

$$\hat{y} = y + \eta. \tag{8}$$

Accordingly, the error function becomes

$$\begin{aligned} \delta &= \ln(\hat{y}) - (a + bx + cx^2) \\ &= \ln(y + \eta) - (a + bx + cx^2). \end{aligned} \tag{9}$$

Expanding it into Taylor series and reasonably omitting the high-order terms, we have

$$\delta \approx \ln(y) - (a + bx + cx^2) + \frac{\eta}{y} \tag{10}$$

so the expectation of δ^2 is

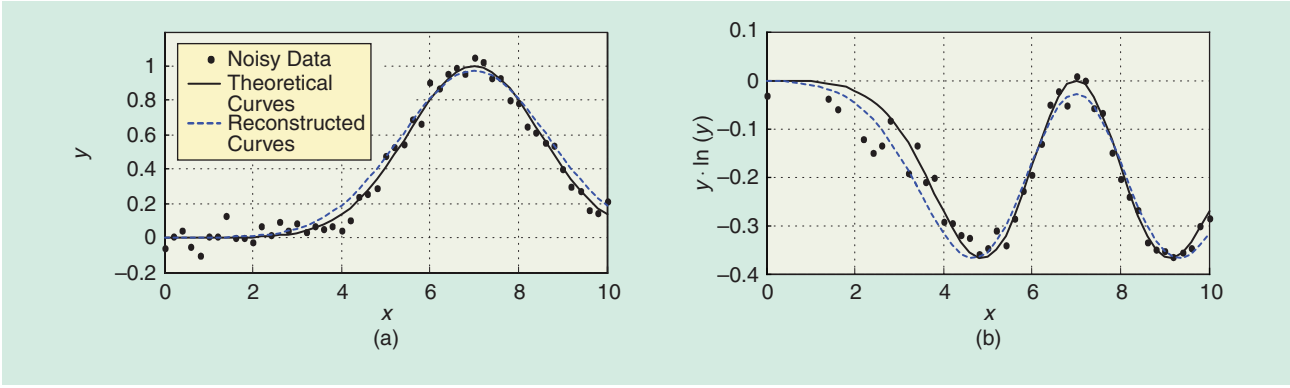
$$E\{\delta^2\} = [\ln(y) - (a + bx + cx^2)]^2 + \frac{\sigma_\eta^2}{y^2}, \tag{11}$$

where σ_η is the standard deviation of the noise.

Noting the second term of (11), if y is very small, the noise at this point will introduce very large errors in the estimates. This fact means that, when we use Caruana's algorithm, the observed data points used in our computations should be limited to those within a narrow range near the peak position of Gaussian curve (for example within the x-interval of $\mu - 2\sigma \le x \le \mu + 2\sigma$), where the Gaussian function has relatively large values. In practice, because the parameters μ and σ are unknown, we usually set a threshold to exclude the data points having very small amplitude values.

If we use only the data points whose y-amplitude values are greater than 0.2, the fitting results are those illustrated with red dashed curves in Figure 1. (The threshold value of 0.2 is determined empirically and must be several times greater than the noise SD value of 0.05.) In this scenario the estimated Gaussian parameters become $A = 0.9639$, $\mu = 6.9806$, and $\sigma = 1.5758$, which are close to the theoretical values.

Even so, the estimation using (11) remains more dependent on the observed points with small values than on those with large ones, and usually a manual intervention has to be performed for



[FIG2] Results of the weighted least squares estimation in the presence of noise.

thresholding the data in advance. We shall solve this problem by employing our proposed weighted least squares algorithm in the next section.

WEIGHTED LEAST SQUARES ESTIMATION

The description of our proposed weighted least squares Gaussian curve fitting algorithm, which overcomes the noise sensitivity of Caruana's algorithm, begins by redefining the error function, using (3), as

$$\begin{aligned}\varepsilon &= y\delta \\ &= y[\ln(y) - (a + bx + cx^2)]\end{aligned} \quad (12)$$

and in the presence of noise

$$\begin{aligned}\varepsilon &= y[\ln(y + \eta) - (a + bx + cx^2)] \\ &\approx y[\ln(y) - (a + bx + cx^2)] + \eta\end{aligned} \quad (13)$$

so the expectation of ε^2 is

$$\begin{aligned}&E\{\varepsilon^2\} \\ &= [y\ln(y) - y(a + bx + cx^2)]^2 + \sigma_\eta^2.\end{aligned} \quad (14)$$

In (14), the influence of y on the second term is removed. Minimizing the sum of ε^2 implies an optimal weighted least squares estimation with the weights equaling y. Differentiating the sum of ε^2 with respect to a, b, and c and setting the resultant expressions to zero yields a linear system of equations of the form

$$\begin{bmatrix} \sum \hat{y}^2 & \sum x\hat{y}^2 & \sum x^2\hat{y}^2 \\ \sum x\hat{y}^2 & \sum x^2\hat{y}^2 & \sum x^3\hat{y}^2 \\ \sum x^2\hat{y}^2 & \sum x^3\hat{y}^2 & \sum x^4\hat{y}^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sum \hat{y}^2 \ln(\hat{y}) \\ \sum x\hat{y}^2 \ln(\hat{y}) \\ \sum x^2\hat{y}^2 \ln(\hat{y}) \end{bmatrix}, \quad (15)$$

In this equation system, because the true values of y are unknown, we have to use $\hat{y}$ instead of y for the weights. Solving (15) for a, b, and c, the parameters of the Gaussian function are further calculated via (5) through (7).

Figure 2 illustrates the performance of our weighted least squares technique. The solid curve and dots in Figure 2(a) denote the theoretical Gaussian function and its sampling data, respectively, which are the same as those in Figure 1(a). The curve reconstructed using our technique is plotted in Figure 2(a) with the blue dashed line. The estimated Gaussian parameters are $A = 0.9689$, $\mu = 7.0184$, and $\sigma = 1.6251$, demonstrating that our technique, which does not exclude small-valued data, is more accurate than Caruana's method.

Of course if we limit our computations by using only the data points having amplitudes greater than 0.2, more accurate results may be obtained, say $A = 0.9807$, $\mu = 7.0114$, and $\sigma = 1.5683$.

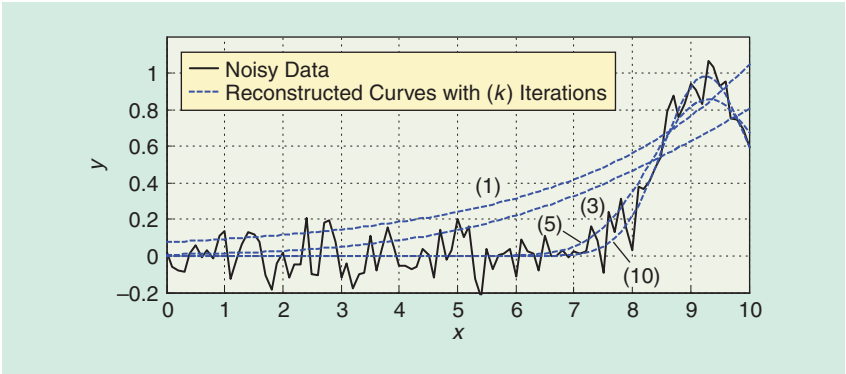
The above results are obtained from one simulation, and a more convincing descriptor is the root-mean-square (RMS) error. This descriptor, as a statistic, is more suitable for describing the behavior of an algorithm in the presence of random noise. As such, we performed 5,000 simulations repeatedly with varying noise (SD = 0.05) and the threshold held at 0.2. With Caruana's algorithm, the RMS errors for the parameters A, μ, and σ are 0.0340, 0.0431, and 0.0779, respectively; whereas, when our proposed technique is used, the RMS errors for the three parameters are reduced to 0.0179, 0.0315, and 0.0554, respectively. These results demonstrate the accuracy of our proposed technique over Caruana's algorithm when fitting a Gaussian peak.

The reason for the improvement of the proposed technique is graphically explained in Figure 2(b), where the vertical axis denotes the logarithms of the Gaussian data multiplied by the weights y. From it, we see that the random fluctuations of the data, induced by the noise, are much more uniform across the full data x-range. Therefore, data having small values, as indicated by (14), will not have an excessively detrimental impact on our estimation results.

ITERATIVE PROCEDURE

Our technique, compared with Caruana's algorithm, is less sensitive to the noise and more accurate in fitting a Gaussian peak. However, if a long tail of the Gaussian function curve is included in the observed data range, the large noise contamination in those data points far away from the peak position still inversely affect our curve fitting accuracies, even lead to failure in the estimation, due to the following reasons. First, (10) cannot fully describe the behaviors of the noise, because the high-order terms of the Taylor series have been omitted in this equation. Second, the true values of y are unknown, so we have to use the noisy values $\hat{y}$ instead of y for the weights in (15). For the observed data points whose values of y are very small, the signal-to-noise ratios may decrease. In other words, the relative difference between $\hat{y}$ and y may be very large, thus inducing considerable error.

We solve this large noise contamination problem by iterating the estimating



[FIG3] **Results of the proposed iterative algorithm.**

technique described by (15) with the weight values being updated in each iteration. The procedure is summarized by

$$\begin{bmatrix} \Sigma y_{(k-1)}^2 & \Sigma x y_{(k-1)}^2 & \Sigma x^2 y_{(k-1)}^2 \\ \Sigma x y_{(k-1)}^2 & \Sigma x^2 y_{(k-1)}^2 & \Sigma x^3 y_{(k-1)}^2 \\ \Sigma x^2 y_{(k-1)}^2 & \Sigma x^3 y_{(k-1)}^2 & \Sigma x^4 y_{(k-1)}^2 \end{bmatrix} \times \begin{bmatrix} a_{(k)} \\ b_{(k)} \\ c_{(k)} \end{bmatrix} = \begin{bmatrix} \Sigma y_{(k-1)}^2 \ln(\hat{y}) \\ \Sigma x y_{(k-1)}^2 \ln(\hat{y}) \\ \Sigma x^2 y_{(k-1)}^2 \ln(\hat{y}) \end{bmatrix}, \quad (16)$$

where

$$y_{(k)} = \begin{cases} \hat{y} & \text{for } k = 0 \\ e^{a_{(k)} + b_{(k)} x + c_{(k)} x^2} & \text{for } k > 0 \end{cases} \quad (17)$$

with the parenthesized subscripts being the iteration indices. Compared with a standard iterative algorithm (e.g., Newton-Raphson) for solving a nonlinear system, our proposed iterative algorithm is computationally much simpler, and the initial guesses for the unknown a, b, and c are not required.

To verify the performance of this iterative procedure, we define a Gaussian function with the parameters $A = 1$, $\mu = 9.2$, and $\sigma = 0.75$. The x-range of observed data points is from zero to ten, and the noisy data is illustrated with the black solid curve in Figure 3, where the SD of the additive noise is 0.1.

This noisy Gaussian function has a narrow peak located near the right edge of the data's x-range. On the left side of the peak there exists a long tail where Gaussian function has very small values and noise is relatively large (the signal-to-noise ratio here is very small). The weighted least squares technique in the previous section is not effective in this situation, but we can succeed by using the iterative procedure that was just introduced. Although the curves in Figure 3 show that the first several iterations cannot produce a satisfactory result, after ten iterations the reconstructed curve fits the theoretical noise-free Gaussian data quite well.

[TABLE 1] ESTIMATED GAUSSIAN PARAMETERS ($A = 1$, $\mu = 9.2$, AND $\sigma = 0.75$).

NUMBER OF ITERATIONS	A	μ	σ
1	0.0648	−6.8010	*j*7.0601
2	0.0270	0.4573	*j*3.4835
3	0.8175	11.0473	2.7769
4	0.8237	9.9600	1.5710
5	0.8890	9.1644	0.9004
6	0.9778	9.1482	0.7564
7	0.9966	9.1468	0.7272
8	0.9990	9.1473	0.7232
9	0.9994	9.1474	0.7226
10	0.9994	9.1474	0.7225

Table 1 lists the estimated Gaussian parameters versus the number of iterations. There we see the convergence of our iterative procedure. In the first two iterations, the large noise in the data makes the estimated σ value to be imaginary, but the final iteration results accurately approximate the theoretical values.

SUMMARY

We proposed an improved technique for estimating the parameters of a Gaussian function from its observed data points. With it, the logarithms of Gaussian data are fitted by using a weighted least squares method derived from a noise model, so that the influences of random fluctuations on the estimation are effectively eliminated. Compared to Caruana's algorithm, our technique is much less sensitive to random noise. Based on our weighted least squares method we also suggested an iterative procedure suitable for reconstructing a Gaussian curve when a long tail of Gaussian curve is included in the observed data points. Because the iterative procedure starts directly from the original data, the initial guesses, which are generally crucial for guaranteeing the convergence of an iterative procedure, are not required, and the implementation of setting a threshold for excluding the small Gaussian data is also unnecessary. Although the techniques proposed in this article focused on fitting a one-dimensional Gaussian function, their principles are easy to extend to multidimensional Gaussian fitting.

ACKNOWLEDGMENTS

The author gratefully appreciates Richard (Rick) Lyons for his suggestions on the content and his assistance with the text of this article. The author also acknowledges the China Scholarship Council and the Mechatronics Engineering Innovation Group Project from Shanghai Education Commission for their support.

AUTHOR

Hongwei Guo (hw-guo@yeah.net) is a professor in the Lab of Applied Optics and Metrology, the Department of Precision Mechanical Engineering at Shanghai University, China.

REFERENCES

[1] Y. Feng, J. Goree, and B. Liu, "Accurate particle position measurement from images," *Rev. Scient. Instrum.*, vol. 78, no. 5, pp. 53–59, May 2007.

[2] R. B. Fisher and D. K. Naidu, "A comparison of algorithms for subpixel peak detection," in *Image technology: Advances in Image Processing, Multimedia and Machine Vision*, J. Sanz, Ed. Berlin: Springer-Verlag, 1996, pp. 385–404.

[3] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed. New York: Cambridge Univ. Press, 2007, pp. 733–836.

[4] R. Caruana, R. Searle, T. Heller, and S. Shupack, "Fast algorithm for the resolution of spectra," *Anal. Chem.*, vol. 58, no. 6, pp. 1162–1167, May 1986.

[5] W. Zimmermann, "Evaluation of photopeaks in scintillation Gamma-ray spectroscopy," *Rev. Scient. Instrum.*, vol. 32, no. 9, pp 1063–1065, Sept. 1961.

[6] R. Abdel-Aal, "Comparison of algorithmic and machine learning approaches for the automatic fitting of Gaussian peaks," *Neural. Comput. Appl.*, vol. 11, no. 1, pp 17–29, June 2002.

[SP]