# CS322 Lecture Notes: Statistics of least squares fitting

Steve Marschner
Cornell University

18-30 April 2007

We discussed least squares fitting, mainly in the linear case, earlier in the course. We used a problem statement like

$$y_i \approx f_i(x_1, \ldots, x_n) \quad ; \quad i = 1, \ldots, m.$$

Often what's really going on is we are fitting some kind of model to some data. In this case we have some function $y(x)$ in mind, the shape of which is controlled by some parameters $a_1, \ldots, a_n$, and our measure of goodness of fit is how well the function $y$ predicts the $y_i$s from the $x_i$s. You can think of $y$ like a box with a bunch of knobs on it, and we're trying to adjust the knobs until the function passes plausibly close to the observed data.

So we can write the goal as

$$y_i \approx y(x_i; a_1, \ldots, a_n) \quad ; \quad i = 1, \ldots, m.$$

This is $m$ equations in $n$ unknowns and we need $m > n$ to have a fitting problem.

In the next few lectures we will look at statistical methods we can use to figure out what, if anything, the results of these fitting calculations are worth.

The overall presentation here follows the two chapters of *Numerical Recipes* fairly closely, with some of the figures shamelessly redrawn from the figures there.

## 1 Example problems

**Fitting polynomials** If the assumption is that $y$ is a polynomial of order $n$ in $x$, the model is:

$$y(x; a_1, \ldots, a_n) = a_1 x^{n-1} + a_2 x^{n-2} + \cdots + a_n$$

This leads to a linear system with a Vandermonde matrix ($m_{ij} = x_j^{n-i}$).

**Harmonic series** If we think the unknown function is periodic with period (w.l.o.g.) $2\pi$, we could use a harmonic series of $n$ sinusoids to represent it. Each sinusoid might be represented by an amplitude $\alpha$ and a phase $\phi$:

$$y(x; \alpha_0, \alpha_1, \phi_1, \ldots, \alpha_n, \phi_n) = \alpha_0 + \sum_{k=0}^{n} \alpha_k \cos(kx - \phi_k)$$

At first blush this seems like it's not linear—and in fact with these parameters it's not: the result is linear in the amplitudes $\alpha_k$, but if we add together two sinusoids the phases $\phi_k$ do not add.

However, this problem can easily be made linear by using different parameters. Look at one term in the sum, and apply a trig identity:

$$\begin{aligned}\alpha_k \cos(kx - \phi_k) &= \alpha_k \cos kx \cos \phi_k + \alpha_k \sin kx \sin \phi_k \\ &= (\alpha_k \cos \phi_k) \cos kx + (\alpha_k \sin \phi_k) \sin kx \\ &= \alpha_k^c \cos kx + \alpha_k^s \sin kx\end{aligned}$$

With the parameters $\alpha_k^c$ and $\alpha_k^s$ it's a linear problem, with matrix rows

$$\begin{bmatrix}\cos x_j & \cdots & \cos nx_j & \sin x_j & \cdots & \sin nx_j & 1\end{bmatrix} \cdot \begin{bmatrix}\alpha_1^c & \cdots & \alpha_n^c & \alpha_1^s & \cdots & \alpha_n^s & \alpha_0\end{bmatrix} \approx y_j.$$

**Exponential decay** Here's another example. We might think we're looking at a process of exponential decay:

$$y(x; a, b) = ae^{bx}.$$

Again the parameter $b$ appears nonlinearly but we can make this into a linear problem by a suitable transformation. This time the transformation is on $y$ rather than $a$: we just take the log of both sides.

$$\ln y = \ln a + bx$$

Now using the parameters $\ln a$ and $b$ we are just fitting a straight line. However, it's important to note that when we reparameterize the right hand side in this way we are changing the fitting criteria: two errors that were equal before the transform to log space will no longer be equal in general. If we look at small errors in the $y$s we can use the derivative $d \ln y/dy = 1/y$ to estimate that if $y_j$ has uncertainty $\sigma_j$ then $\ln y_j$ has uncertainty $\sigma_j / y_j$.

These few examples serve to illustrate that more problems can be handled by linear least squares than you might think. Of course, many problems are irrevocably nonlinear.

## 2 Goals of statistical analysis

The two basic questions about fitting results that we'll look for ways to answer are

- How well does the model fit the data?
- What is the accuracy of the parameters found by the fitting process?

The best way to go about answering these questions is armed with estimates of the uncertainties in the data values $y_i$. (For this course we'll assume the $x_i$s are accurate.) Then we can think of fitting problems as statistical problems: we know the system is overdetermined and we won't expect an exact match. We would like to attribute the error to noise: we assume we have the correct mode and that the residual is simply caused by random errors in the data. In this framework we can rephrase the two questions as:

- Can the residual be plausibly explained by random error?
- What range of parameter values could have resulted from different random errors in the data?

The two approaches to finding the answer that we'll explore are

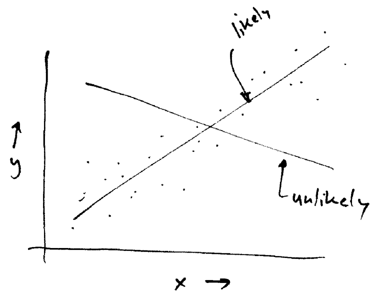
- Monte Carlo simulation. This simple brute force approach works for all kinds of problems and is easy to understand but slow and limited in some ways.
- Statistics of $\chi^2$ distributions. Some results in traditional statistics can be used to directly compute quality and confidence measures, as long as the fitting process is linear and the errors are Gaussian.

## 3 Continuous probability primer

We'll be using continuous probability to talk about the distributions of data and parameter values. Since only discrete probability is included in the prerequisites, a quick briefing/review is in order. We won't be doing any heavy-duty computations with continuous probabilities but you should know the basic definitions to fully make sense of what the numbers mean.

[This section to be expanded.]

## 4 Least squares as maximum likelihood estimation



Some parameter values are "likely" to be right, because the model passes plausibly close to the data points, relative to the error in the data. Others aren't: they are so far away from some points that it's extremely unlikely that the data is off by that much.

Notice I didn't say some parameter values are "probable" and others aren't. Fundamentally it doesn't really make sense to talk about probabilities of different parameter values, because it's not the parameters that are uncertain. The real system has some (fixed) parameter values—it's just that we don't know them. The randomness is in the data values, which are contaminated by noise that we assume comes from some random process.

In other words the probability space of the problem is the data values, or more specifically the noise in the data.

We can still formulate the idea of some models being statistically preferable over others. Rather than the probability of the model given the data, we talk about probability of the data given the model. Instead of asking the question "which parameters are most probable?" we ash the question

> Which values of the parameters, if they were the true parameters, would lead to the highest probability of having observed the data that we did?

Remember that we're assuming we know what the random noise-generating process is. That is, for any set of parameters, there is some probability distribution that describes the data that we would measure from a system with those parameters. If we evaluate that distribution for the data $y_j$ that we actually did observe, we get a probability of observing that data. If the probability is low, it's not likely we have the right parameters. So we call this "backwards" probability the "likelihood" of the parameters.

Finding the parameters by maximizing the likelihood is called **maximum likelihood estimation**, and leads to the **maximum likelihood estimate**. Both are abbreviated MLE.

Maximum likelihood estimation has a close connection to least squares fitting—in fact, the two methods are the same *for independent Gaussian errors.* This is a prime statistical excuse (er, I mean motivation) for using the mathematically convenient least squares fitting method.

The connection is simple to see. Suppose the errors are independent and normally distributed with identical variance. This means the probability for one data point is:

$$p(y_j) = \exp\left[-\frac{1}{2}\left(\frac{y_j - y(x_j)}{\sigma}\right)^2\right]$$

and the probabilities of the data points multiply (since they're independent):

$$p(\mathbf{y}) = \prod_{j=1} m \exp\left[-\frac{1}{2}\left(\frac{y_j - y(x_j)}{\sigma}\right)^2\right]$$

To get the MLE we need the $\mathbf{y}$ that maximizes $p(\mathbf{y})$. This is equivalent to minimizing

$$-\ln p(\mathbf{y}) = \frac{1}{2\sigma^2}\sum_{j=1}^{m}\big(y_j - y(x_j)\big)^2$$

The sum on the right hand side is just the familiar least-squares residual. So the $\mathbf{y}$ that maximizes likelihood is the $\mathbf{y}$ that minimizes the least-squares residual.

If the errors are independent and Gaussian but each has its own standard deviation $\sigma_j$, we end up in nearly the same place. Losing the constant factor of one-half, we get a standard formula that gets the fancy name "chi squared":

$$\chi^2 = \sum_{j=1}^{m}\left(\frac{y_j - y(x_j;\mathbf{a})}{\sigma_j}\right)^2$$

[**Caution:** Are the errors in your measurements really normally distributed? A couple of examples: image intensities in digital cameras come from a counting process and are distributed according to a Poisson, not Gaussian, distribution. Measurements made using a scale like a ruler are very unlikely to be off by more than the space between tics: the error is more uniform that Gaussian. Also, you might have outliers: points that are mixed in that are generated by a different process (for example, caused by human error, static discharge, automatically detecting and measuring the wrong feature, etc.).

With this in mind we'll happily forge ahead anyway. In many (most?) problems the normal error assumption in practice leads to useful estimates of parameters and uncertainties; we just need to remember to keep the (in)validity of this assumption in the backs of our minds when interpreting the results.]

We can answer our quality and certainty questions by looking at the statistical distribution of $\chi^2$: quality of fit is related to the distribution of the residual $\chi^2$, and confidence in the parameters is related to the distribution of fitted parameters.

For quality of fit: suppose we have a good model (that is, it fits the underlying "true" data before it was corrupted by noise). The residual of the fit depends on what noise was added to the data (in fact, it depends only on the component of the noise that lies in the range complement of $A$), so the probability distribution of the noise induces a probability distribution on the residual. Using this distribution we can ask, "what is the probability that our residual would be this high just by chance?" If the answer is extremely low, it's unlikely that our model really fits the data—that is, random error alone can't explain why we are so far off. Alternatively one can ask "what is the probability that our residual would be this low just by chance?" (This is one minus the other probability.) If the answer is extremely low, it suggests something is wrong: probably we overestimated the uncertainties.

For confidence in parameters: The fitted parameters also depend on the noise in the data (in fact, they depend only on the component of noise that lies in the range of $A$). We can express the degree of certainty by quoting a *confidence region*: "There's a 90% probability that the parameters lie in this region." That is to say, if we repeated the experiment many times, with different noise each time, the parameters would be in the confidence region 9 times out of 10. You can replace the probability 0.9 with any number that makes sense for a given application.

## 5 Monte Carlo approach

Here is our mental model for where the data comes from:

$$\begin{tikzcd}
 & \eta \arrow[r] & y_0 \arrow[r, "\text{fit}"] & a_0 & \text{actually happened.} \arrow[l] \\
 & \eta_1 \arrow[r, dashed] & y_1 \arrow[r, dashed] & a_1 & \\
a_{\text{true}} \arrow[uur] \arrow[ur, dashed] \arrow[dr, dashed] & & & & \text{Could also have happened.} \\
 & \eta_2 \arrow[r, dashed] & y_2 \arrow[r, dashed] & a_2 & \\
 & & \vdots & &
\end{tikzcd}$$

The $\{\mathbf{a}_i\}$ have some distribution, and that distribution answers questions about how certain the parameters are.

What we really want is the distribution of $\mathbf{a}_i - \mathbf{a}_{\text{true}}$ because that is the error that actually matters. Unfortunately we don't have access to $\mathbf{a}_{\text{true}}$. A principle that lets us continue is that the distribution of errors in the real problem is the same as the distribution of errors that would result if our parameter estimate $\mathbf{a}_0$ was the true value. That is, the effects of errors aren't changing so fast that errors added to the true model have noticeably different behavior than errors added to our estimated model.

This leads to a simple brute force approach: pretend that the parameters $\mathbf{a}_0$ describe the real system, and generate $N$ synthetic data sets by adding randomly generated noise according to our model of the uncertainty:

$$\mathbf{y}_i^S = y(\mathbf{x}, \mathbf{a}_0) + \eta_i$$

where the noise $\eta_i$ is generated from some model of the uncertainty. We might use $N = 10^5$ or more, if the fitting process is pretty fast. Then we fit parameters to all those synthetic datasets and look at the distribution of the resulting parameter vectors $\mathbf{a}_i^S$:

$$\begin{tikzcd}
 & & \eta_1 \arrow[r] & \underline{y}_1^s \arrow[r, "fit"] & \underline{a}_1^s \\
\underline{y}_0 \arrow[r, "fit"] & \underline{a}_0 \arrow[ur, bend left] \arrow[r] \arrow[dr] & \eta_2 \arrow[r] & \underline{y}_2^s \arrow[r] & \underline{a}_2^s \\
 & & \eta_3 \arrow[r] & \underline{y}_3^s \arrow[r] & \underline{a}_3^s \\
 & & & \vdots &
\end{tikzcd}$$

The idea is that $\{\mathbf{a}_i^S\}$ is distributed around $\mathbf{a}_0$ in the same way as $\{\mathbf{a}_i\}$ is distributed around $\mathbf{a}_{\text{true}}$, so we can answer our questions about the distribution of $\mathbf{a}_0$ over all possible datasets by instead asking questions about the distribution of $\mathbf{a}_i$ that we computed.

Following the previous section, we can compute the distribution of residuals that we need to evaluate fit quality just by counting:

$$\Pr\{\chi^2 < x\} \approx \frac{\#(\chi^2(\mathbf{a}_i^S) < x)}{N}$$

and from these numbers we can estimate whether our fit is improbably good or bad.

Also following the previous section, we can estimate parameter confidence by simply looking at the actual distribution of the $\mathbf{a}_i^S$. For example, we can generate single-parameter confidence intervals simply by finding an interval that contains the requisite fraction of the total points. For example, you could find a 90% confidence interval by reporting the 5th and 95th percentile of that parameter's values.

When correlations between parameters are important, it's often more useful to quote a confidence ellipse (or ellipsoid for more than 2 variables) rather than separate confidence intervals. In this case our job is to find an ellipse that contains, say, 90% of the points. We'd like the ellipse to fit the distribution so that it is as small as possible; we can do this using principal components. Just find the SVD of the point cloud (don't forget to subtract the mean first) and the singular vectors and values give you axis directions and axis length ratios for ellipses that fit the shape of the distribution. Expressing the points in terms of the singular vectors, so that your ellipses are circles, makes it easy to define the ellipse that contains the required fraction of the points.

## 6 Gaussian distributions

Before I talk about the analytic methods for finding the distributions of residuals and of parameter values, I will review Gaussians and their convenient properties.

I'll talk about Gaussians in the linear-transformation oriented way: Start with a unit (by which I mean unit variance, which is the same as unit standard deviation) Gaussian in $I\!R^m$:

$$g(\mathbf{x}) = \exp\Big(-\sum_i \frac{x_i^2}{2}\Big) = \exp\Big(-\frac{\mathbf{x}\cdot\mathbf{x}}{2}\Big) = \exp(-\|\mathbf{x}\|^2/2)$$

This function is both separable (it is the product of a function of each variable) and radially symmetric (it is a function of radius; that is, of the norm of $\mathbf{x}$).

Note that if I hold $n$ of the $m$ $x$s constant and look at the distribution as a function of the other $m-n$ variables, I see an m-n dimensional Gaussian. In probability-speak, all the conditional distributions of the Gaussian are also Gaussians. Also, if I integrate along any axis, thereby eliminating one of the variables:

$$g_m(x_2,\ldots,x_m) = \int_{-\infty}^{\infty} g(\mathbf{x})\,dx_1$$

the resulting function $g_m$ is an $(m-1)$-dimensional Gaussian. That is, all the marginal distributions of the Gaussian are also Gaussians.

Now suppose we draw points from this distribution and map them through a linear transformation $\mathbf{X}$:

$$\mathbf{y} = \mathbf{X}\mathbf{x}$$

The probability density of $\mathbf{y}$ is $g'(\mathbf{y}) = g(\mathbf{X}^{-1}\mathbf{y})/|\det\mathbf{X}|$. It's easy to come up with confidence regions: find a (circular) confidence region for the distribution of $\mathbf{x}$, and map it through $\mathbf{X}$ to get an ellipse:

The SVD of $\mathbf{X}$, $\mathbf{U}\Sigma\mathbf{V}^T$, can give us the principal axes of the ellipse (or ellipsoid). If the confidence region for $g(\mathbf{x})$ has radius $r$, then the confidence region for $\mathbf{y}$ has axes $\{r\sigma_i u_i\}$ where the $\sigma$s and $u$s are the singular values and left singular vectors of $X$.

These are all straightforward properties of the Gaussian distribution that follow from its algebraic form.

The one final mathematical tool we'll need is one to tell us how large a confidence region we need to use. It's sufficient to be able to find these regions for a unit Gaussian, since we can always transform them to deal with other Gaussians.

The question we need to answer is: "What is the probability that a point drawn from a unit Gaussian is within a distance $r$ from the origin?" The answer to this question tells us what confidence we can quote for a sphere of radius $r$ around the origin.

There's a standard probability distribution designed to answer this question, called the $\chi^2$ distribution because of its association with $\chi^2$ fitting. I actually wish it had a different name, because it's really a simple idea that makes perfect sense separately from the model-fitting context.

If we draw points randomly from a $\nu$-dimensional Gaussian distribution, and look at the distribution of the squared norms of the points (that is, look at the distribution of the random variable $X = \|\mathbf{x}\|^2$ over a Gaussian probability space), the distribution is the $\chi^2$ distribution for $\nu$ degrees of freedom. Another way of saying the same thing is that the $\chi^2$ distribution is the distribution of the sum of $\nu$ independent gaussian-distributed random variables.

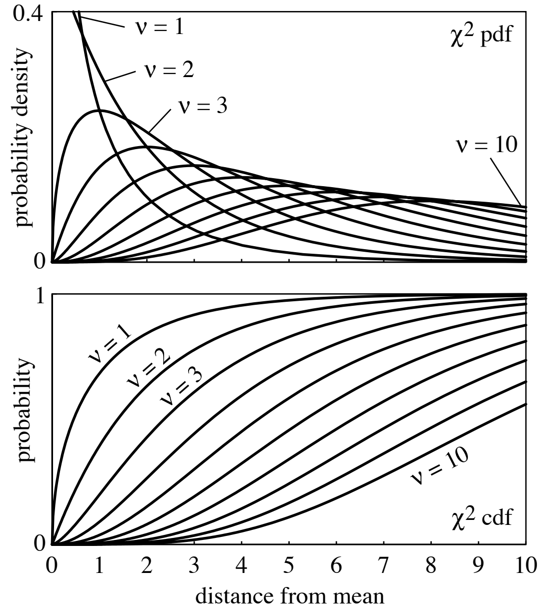


Figure 1: The probability density function $p_{\chi^2}(r|\nu)$ (top) and cumulative distribution function $P_{\chi^2}(r|\nu)$ (bottom) for the $\chi^2$ distribution, for degrees of freedom $\nu$ ranging from 1 to 10.

We can explore the $\chi^2$ distribution using Monte Carlo very simply by adding up $\nu$ gaussian random numbers a million times and making a histogram. Here is a Matlab transcript of this demo:

```
figure
N = 1e6;
dx = 0.1; xs = (0:dx:10) + dx/2;
plot(xs, histc(randn(1,N).^2, 0:dx:10)/N/dx, '-')
hold on
plot(xs, histc(sum(randn(2,N).^2), 0:dx:10)/N/dx, '-')
axis([0 10 0 0.4])
plot(xs, histc(sum(randn(3,N).^2), 0:dx:10)/N/dx, '-')
plot(xs, histc(sum(randn(4,N).^2), 0:dx:10)/N/dx, '-')
plot(xs, histc(sum(randn(5,N).^2), 0:dx:10)/N/dx, '-')
plot(xs, histc(sum(randn(6,N).^2), 0:dx:10)/N/dx, '-')
plot(xs, histc(sum(randn(7,N).^2), 0:dx:10)/N/dx, '-')
plot(xs, histc(sum(randn(8,N).^2), 0:dx:10)/N/dx, '-')
```

I'll denote the $\chi^2$ probability density function by $p_{\chi^2}$ and its cumulative distribution function $P_{\chi^2}$:

$$\left.\begin{aligned} p_{\chi^2}(r^2|\nu) &= \text{pdf for } \|\mathbf{x}\|^2 \text{ at } r^2 \\ P_{\chi^2}(r^2|\nu) &= \Pr\{\|\mathbf{x}\|^2 < r^2\} \end{aligned}\right\} \text{ for Gaussian-distributed } \mathbf{x} \text{ in } \mathbb{R}^\nu.$$

These functions don't really have closed forms, but there are standard functions available for computing their values (in Matlab, they are `chi2pdf` and `chi2cdf`). They compute the same thing as the Monte Carlo experiment but much more quickly and accurately:

```
for j = 1:8
   fplot(@(x) chi2pdf(x, j), [0 10], 'r-')
end
```

The $\chi^2$ distribution and its cumulative probability (the probability of $\chi^2$ being less than a given value) are shown in Figure 1.

The cumulative probability $P_{\chi^2}$ is exactly what we need to answer questions about probability of having residuals or parameter values in certain ranges, which I'll explore in the next section.

## 7 Fitting statistics for Gaussian errors

In the section on Monte Carlo, we explored *what* probabilistic questions we need to answer in order to evaluate the meaningfulness of fitting results. In the section on Gaussians, we developed tools we can use to answer these questions for the special case of errors that happen to come in a Gaussian distribution. The reasoning is exactly the same as before, so I'll just explain how we compute the relevant statistics for Gaussians.

The first statistic we examined was the distribution of residual errors over the distribution of noise in the data. The data is distributed about the (unknown) $\mathbf{y}_{\text{true}}$ in an $m$-dimensional Gaussian, and the squared magnitude of the noise is distributed according to $\chi^2$ with $m$ degrees of freedom. There's a subtlety here, though: the residual has fewer degrees of freedom, because, unlike the real answer, the fitted model $\mathbf{A}\mathbf{a}_0$ always matches the data $\mathbf{y}_0$ in all the directions that are in the range of $\mathbf{A}$. This means the residual is normally distributed in the range complement of $\mathbf{A}$, which has $(m - n)$ dimensions. So the residual of a $\chi^2$ fit is distributed according to the $\chi^2$ distribution with $\nu = m - n$.

The second statistic was the distribution of parameter values around the fitted parameters $\mathbf{a}_0$. In this case, the parameters are projected *into* the range of $\mathbf{A}$, which means the relevant $\chi^2$ distribution has $n$ degrees of freedom. Using the SVD $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$, we can think of the process that maps $\mathbf{y}$ to $\mathbf{a}$ as a projection from $m$ to $n$ dimensions followed by a transformation in $n$ dimensions:

$$\begin{aligned}\mathbf{a} &= \mathbf{V}\Sigma_1^{-1}\mathbf{U}_1^T\mathbf{y} \\ &= (\mathbf{V}\Sigma_1^{-1})(\mathbf{U}_1^T\mathbf{y}) \qquad (1)\end{aligned}$$

The projection by $U_1$ throws out $m - n$ dimensions, leaving $\mathbf{U}_1^T\mathbf{y}$ distributed as a Gaussian in $n$ dimensions. (This is one of the special properties of Gaussians: if you look at some of the variables, integrating the others out, you still see a Gaussian.) We can compute (spherical) confidence intervals for $\mathbf{U}_1^T\mathbf{y}$ using $P_{\chi^2}(\cdot, n)$ and then transform them by $\mathbf{V}\Sigma_1^{-1}$ to get the confidence regions for $\mathbf{a}$.

We also can compute confidence intervals for the parameters separately. In the Monte Carlo technique we would just ignore the other parameters when computing the confidence region. Now we have to be a little more careful to keep track of what distribution we are looking at. Suppose we are interested in the parameter $a_k$. Looking at just the $k$-th row of (1), we see that it is the product of the $k$-th row of $\mathbf{V}\Sigma_1^{-1}$ (call it $v'_k$) with the normally distributed points $\mathbf{U}_1^T\mathbf{y}$. This amounts to a scaled projection of that (unit variance) Gaussian distribution onto a single axis, so we end up with the parameter distributed according to a $p_{\chi^2}(\cdot, 1)$, but scaled by the length $\|v'_k\|$.

## Sources

- Press, Flannery, Teukolsky, and Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Second Edition. Cambridge University Press, 1992. Chapters 14–15.