Statistics, Data Analysis, and Machine Learning for Physicists
Timothy Brandt
Spring 2020

Lecture 5

In this lecture we'll briefly discuss numerical integration, and then apply that discussion to the calculation of confidence intervals when explicitly integrating a likelihood function. Numerical integration is the subject of Chapter 4 of Numerical Recipes. We will skip most of this chapter, since it is beyond the scope of this course and I have generally found it to be of limited usefulness in my work.

In one dimension, a numerical integral is an evaluation of

$$\int_a^b f(x)dx. \tag{1}$$

The integral is defined as a Riemann sum,

$$\int_a^b f(x)dx = \lim_{N\to\infty} (b-a) \sum_{i=1}^{N} \frac{f(a+(b-a)(i/N))}{N}. \tag{2}$$

You can evaluate this directly by progressively increasing the number of points at which you evaluate the function. It converges relatively slowly; its error scales as

$$\int_a^b f(x)dx = \frac{b-a}{N} \sum_{i=1}^{N} f\left(a + i\frac{b-a}{N}\right) + \mathcal{O}(f'/N). \tag{3}$$

Defining

$$h = \frac{b-a}{N}, \tag{4}$$

this is written as

$$\int_a^b f(x)dx = h \sum_{i=1}^{N} f\left(a+ih\right) + \mathcal{O}(hf'). \tag{5}$$

Twice the precision (half the error) requires twice the number of data points. You can do better, as you probably know from introductory calculus, by approximating the function as piecewise linear, piecewise quadratic, etc., and doing the integrals. This is the origin of the trapezoidal rule, Simpson's rule, etc. The trapezoidal rule, for example, is given by

$$\int_a^b f(x)dx = h\left(\frac{1}{2}f(a) + f(a+h) + \ldots + f(b-h) + \frac{1}{2}f(b)\right) + \mathcal{O}(h^2 f''). \tag{6}$$

The integral is now exact for a linear function; the error scales with the second derivative and decreases with stepsize as h^2 (assuming the second-order terms dominate the residual–if your function is not smooth then your convergence may not be $\sim h^2$). Twice the number of points gets me four times the precision.

In calculus, you probably next learned Simpson's rule, and maybe even a higher-order quadrature scheme. I am not going to do that here; I will stop at the trapezoidal rule. I will instead show you a trick that lets you make the trapezoidal rule converge at any rate you desire.

Equation (7) gives the error term as being second order in h, where there is a constant factor that you could calculate and the second derivative evaluated somewhere in the interval (a,b). Let's assume that I can simply write

$$\int_a^b f(x)dx = h\left(\frac{1}{2}f(a) + f(a+h) + \ldots + f(b-h) + \frac{1}{2}f(b)\right) + Ch^2 \tag{7}$$

where C is roughly constant for small h. This will work for well-behaved functions with sufficiently small h though I will make no attempt at a proof. Let's use this to get a better estimate of the integral. I will denote the trapezoidal rule with step size h as $T(h)$, so that

$$T(h) \approx \int_a^b f(x)dx - Ch^2, \tag{8}$$

$$T(h/2) \approx \int_a^b f(x)dx - C(h/2)^2. \tag{9}$$

If I want to cancel the C and solve these equations simultaneously, I get

$$\int_a^b f(x)dx \approx \frac{4}{3}T(h/2) - \frac{1}{3}T(h). \tag{10}$$

You can plug in for yourself and verify that this is *precisely* Simpson's rule. Now, it turns out that all of the error terms in Equation (7) that have odd powers of h vanish, so only even powers remain. Suppose I want an integration scheme with an error term that goes as h^6 (Simpson's rule goes as h^4). I can then just solve the system of equations

$$T(h) \approx \int_a^b f(x)dx - c_1h^2 - c_2h^4 \tag{11}$$

$$T(h/2) \approx \int_a^b f(x)dx - c_1(h/2)^2 - c_2(h/2)^4, \tag{12}$$

$$T(h/4) \approx \int_a^b f(x)dx - c_1(h/4)^2 - c_2(h/4)^4. \tag{13}$$

This is a linear system, so just compute the trapezoidal rule three times (you can even use the fact that you have already computed half of the points in the previous computation to save yourself half the time). Then solve the linear system, and presto, you have an integration scheme that is good to fifth order. This trick is called *Richardson extrapolation* or *Romberg integration* and it works extremely well for smooth functions.

I won't say much about improper integrals, even though the integral in Problem 2 of your homework is improper (one or both limits is ∞). You can remove the improper nature by changing variables, e.g.

$$\int_0^\infty e^{-x} = \int_0^1 e^{-x} + \int_1^0 e^{-1/x}d(1/x) \tag{14}$$

$$= \int_0^1 e^{-x} - \int_0^1 e^{-1/x}\frac{dx}{x^2}, \tag{15}$$

or you can just take your limits to be far enough along that the function is very small. Either method will work for your homework; I will suggest the second option for simplicity. As one additional remark, suppose that you can't evaluate your function at an endpoint (this is indeed the case for the improper integral given above). Try replacing Equation (7) with

$$\int_a^b f(x)dx = h\left(f(a+h/2) + f(a+3h/2) + \ldots + f(b-h/2)\right) + Ch^2 \tag{16}$$

and all of the preceding discussion is identical–this is just as good a building block for Richardson extrapolation to a quadrature scheme of arbitrary order. Note that you can't take twice as many steps using Equation (16) and still use your previous result, but you can take three times as many steps and reuse computations. So, you could build your integration scheme out of things like $T'(h)$, $T'(h/3)$, etc. if you want to save that little bit of effort.

Numerical Recipes also includes a discussion of integration where you can choose where to evaluate your function. In the cases above, we could get as much as n-th order accuracy with $n+1$ function evaluations.

If we give ourselves the freedom to choose the points at which to evaluate the function, we can actually get *twice* the convergence order. However, accuracy in the result requires the function you are actually integrating to be well-approximated by a polynomial. There are even further tricks and this is all very neat, but I won't spend additional time on it and I just refer you to the book if you are interested.

The last topic for us today is of special relevance to this homework and to likelihood functions in general: integration in multiple dimensions. This is a very difficult problem. If I want to take a step size h corresponding to N steps in one dimension, a quadrature scheme of the same order in d dimensions requires N^d steps. If $d = 5$, for example, this could be prohibitively expensive. Techniques like Markov Chain Monte Carlo are basically ways to evaluate horrible multidimensional integrals. For Problem 2 of the homework, $d = 2$ and it isn't all that bad. You can define your function on a grid of points and integrate in one dimension, then the other, using Richardson extrapolation of the trapezoidal rule in each case. You could even just define your function over a grid of points and do a straight Riemann sum, taking a very large number of points to get convergence (modern computers are fast enough to let you get away with this lazy approach, at least for $d = 2$).

The task for computing confidence intervals then comes down to the following:

1. First, normalize your likelihood to get a probability distribution. This means calculating

$$\int \mathcal{L}(\mathbf{x})\,\mathrm{d}^n\mathbf{x}, \tag{17}$$

   where the integral is, in general, over multiple dimensions. The probability distribution of the parameters $\mathbf{x}$ is then

$$p(\mathbf{x}) = \mathcal{L}(\mathbf{x}) \left(\int \mathcal{L}(\mathbf{x})\,\mathrm{d}^n\mathbf{x} \right)^{-1}. \tag{18}$$

2. Compute the one-dimensional function $q(t)$ defined by

$$q(t) = \int_{p(\mathbf{x}) \le t} p(\mathbf{x})\,\mathrm{d}^n\mathbf{x} = \int p(\mathbf{x}) \cdot (p(\mathbf{x}) \le t)\,\mathrm{d}^n\mathbf{x} \tag{19}$$

   where I am using $(p(\mathbf{x}) \le t)$ to denote a function that equals 1 if $p(\mathbf{x}) \le t$ and 0 otherwise. With this convention, $q(0) = 0$ and $q(p_{\max}) = 1$. The function q is monotonically increasing and represents the integrated probability below some threshold in the probability density. If this number is small, then most of the probability is at higher probability densities and only a small amount is at lower probability densities.

3. Find, e.g., the value of t for which $q(t) = 1 - 0.683 = 0.317$ (for a 1σ equivalent confidence interval). For a 3σ interval, find t for which $q(t) = 1 - 0.9973 = 0.0027$. The resulting t values are where you draw your contours.

Task 1 is straightforward enough (if not necessarily easy). Task 2 is also straightforward given a value of t, but Task 3 requires inverting $q(t)$. The function $q(t)$ is monotonically increasing with t, so the existence of an inverse isn't a problem. To perform the inverse, first compute $q(t)$ at a range of sample t. Then use spline interpolation, but when you do this, pass $q(t)$ to the function as your independent variable and t as your dependent variable. You'll get a spline fit to your inverse as long as you have sampled t and $q(t)$ reasonably well around the contours of interest.

To make this latter part explicit, use the χ^2 thresholds from all of the Gaussian assumptions to make a reasonably well-sampled set of t values. For each one, compute $q(t)$. Then use interpolation as follows:

```
delta_chisq = numpy.arange(1, 20, 0.5)                # use your discretion here
tvals = numpy.amax(likelihood)*numpy.exp(-delta_chisq/2)
q_tvals = integrate(likelihood*(likelihood < tvals))  # note that this is pseudo-code
qinv = interpolate.interp1d(q_tvals, tvals, kind='cubic')
sigvals = numpy.asarray([1, 2, 3])                    # sigma thresholds to use
pvals = scipy.special.erfc(sigvals/numpy.sqrt(2))     # probabilities corresponding to sigma
thresholds = qinv(pvals)                              # and we're done!
```

If you have a way to explicitly integrate your likelihood function, then the code above will get you your thresholds for your confidence intervals. Note that the integral in the third line above is very much pseudo-code–you can't just copy that line into python, but will have to pass the likelihood and tvals to a function that you write to actually do the integration. For this function, since you have a threshold and therefore complicated limits (or equivalently a function that isn't smooth because of its endpoints), there is no point to very high-order integration. Don't try to get fancy; it won't work. Just use the trapezoidal rule, maybe with Richardson extrapolation to get two extra orders (i.e. Simpson's rule), and then patiently wait for convergence. You can check for convergence of the integral when halving the step size changes the result by less than some threshold (say, a factor of 10^{-5}).

Another way to do this with a simple Riemann sum is to define $\mathcal{L}$ over a very fine grid in the two parameters of interest. You will then have a 2-D array of likelihoods, which you can sort using

```
likelihood_sorted = numpy.sort(likelihood*dxdy, axis=None)
integrated_likelihood = numpy.cumsum(likelihood_sorted)
integrated_probability = integrated_likelihood/integrated_likelihood[-1]
```

In this I have not assumed that the 2-D volume element $dxdy$ is constant; if it is, you can omit it. You can then apply your spline fit directly to the x and y represented by the integrated probability ($q(t)$) and sorted likelihood (t). If you do this, verify that it has converged by taking a finer grid and/or one extending farther out and verifying that you get the same thresholds.