Statistics, Data Analysis, and Machine Learning for Physicists

Timothy Brandt
Spring 2020

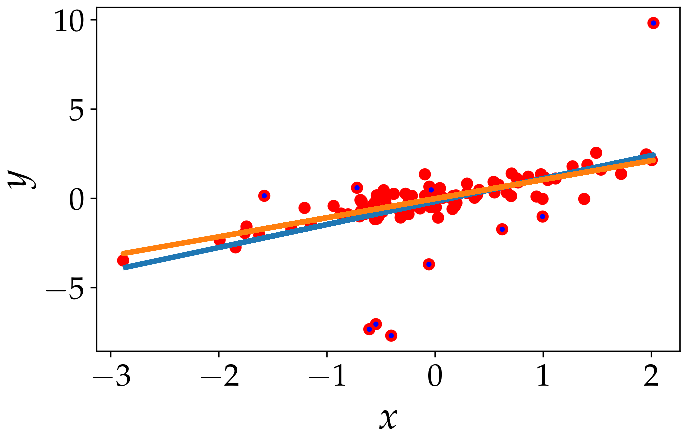
Lecture 10

In this class we'll talk a bit about what you can do if we start to break our assumptions about Gaussianity. This typically comes in a couple of flavors:

1. You don't trust that your uncertainties are Gaussian (or Poissonian) at all.
2. You think your uncertainties are reasonably Gaussian, but are prone to outliers.

The first problem is very hard. If your statistics are either Gaussian or Poissonian, then you can write down a likelihood function pretty straightforwardly. If they are from some other distribution that you know, then perhaps you can still write down a likelihood function. You probably won't be able to apply the linear least-squares technique to maximize or integrate it, though. You might have to turn to something like Markov Chain Monte Carlo, which we will cover soon.

I'll talk mostly about the second problem first: your data have outliers. If you know which points are outliers before you start your analysis, you can just throw them out. Here I am assuming that you don't have that luxury. To illustrate this, I have generated 100 points from a very simple model $y = x$, and added some measurement uncertainty to the y values. I then added a much larger random error, drawn from a uniform distribution, to 10% of the points. These are outliers in the sense that the error model is incorrect. They might not be outliers in the sense that they might not deviate all that badly from the fit. The plot below shows these points with red dots, the points to which I added extra uncertainty are shown with small blue dots in the center. The blue line is fit to all points, including outliers; it has a poor χ^2 and its slope is many sigma away from the truth. The orange line is fit only to points without blue circles.



The key problem here is that *I don't know which points to throw out.* I cannot tell which points should have blue circles associated with them. Some outliers are obvious, but other aren't. And then there's the danger of cherry picking. If you stare at that plot, you can probably see that a few of the points are outliers and you can mask them by hand. But this is a dangerous approach that can easily go too far. If you are throwing out every point that doesn't fit your model, of course you are going to find that your model is a good fit! This can be, and often is, a bit more subtle. Maybe you take a very close look at every point that seems to be far from your fit in search of a reason to throw it out, but you don't look so closely at the rest. If you look hard enough at any data point, you can probably find a reason to toss it out. We want something a bit more systematic. In this class, we'll go through a couple of options. All of these are relatively common practice, and you will probably encounter them in analyses that you read. Some of these are also at least somewhat appropriate to the case where your errors are not very Gaussian.

We'll start in the same place as before, with χ^2. I'll assume a diagonal covariance matrix here. The case of a nondiagonal covariance matrix with outliers is harder: you are now coupling highly problematic error estimates between points. One outlier can then mess up many measurements. You can excise a measurement from your data set, removing that row and column from the covariance matrix, but it's a bit harder. The following techniques are *a lot* easier to apply to the case of a diagonal covariance matrix, with

$$\chi^2 = \sum_i \frac{\left(\mathrm{data}_i - \mathrm{model}_i\right)^2}{\sigma_i^2}. \tag{1}$$

In the following discussion, I'll assume that your covariance matrix is diagonal, i.e., that your measurements are independent.

The first approach is known as *iteratively reweighted least squares*, or IRLS. I can write ordinary χ^2 as

$$\chi^2 = \sum_i w_i \left(\mathrm{data}_i - \mathrm{model}_i\right)^2 \tag{2}$$

where the weights w_i are the inverse variances. You might guess from the name that IRLS adjusts these weights after each iteration of a χ^2 fit. If the weights change, then the model that minimizes χ^2 will also, in general, change.

I'll mention the simplest way of adjusting the weights: you can set them to zero for points that you think are outliers. In other words, after each iteration, your weights might look like

$$w_i = \begin{cases} 1/\sigma_i^2 & \text{if } \ |\mathrm{data}_i - \mathrm{model}_i| < n\sigma_i \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

This is pretty common and also goes by the name *sigma clipping.* You throw out points that are more than n-σ discrepant with the fit. This hard cut isn't ideal, but this clipping is easy to implement and can take care of rare but catastrophic outliers. Implementing it in code is pretty straightforward. At each iteration, you could set a mask

```
    mask = (data - model)**2*ivar < nsig**2
```

and then do your minimization using `ivar*mask` as your weights. The array `mask` is `True` (or `1`) when the data and model are within `nsig` of one another, and `False` (or `0`) otherwise. You can stop when your boolean array of masks stops changing–at this point the iterative reweighting has converged.

I'll go over one other way of handling outliers: a mixture model. This will be better in the sense of robustness to more abundant outliers and continuity in its treatment of outliers (the IRLS masking suggested above is discontinuous as a point crosses the magical $n\sigma$ boundary). It will be worse in computational difficulty.

The approach is the following. We assume that each measurement could either be good, i.e, drawn from the distribution that we assume the errors to come from, or bad, i.e., drawn from some other and presumably much broader, distribution. We assume that we know the proper distribution well and we crudely approximate the much broader distribution, for example as a Gaussian with ten times the width. The likelihood for this single measurement y_i given a model x_i is then

$$\mathcal{L}_i = p(\mathrm{good})\frac{1}{\sqrt{2\pi\sigma_i^2}}\exp\left[-\frac{(x_i - y_i)^2}{2\sigma_i^2}\right] + p(\mathrm{bad})p_{\mathrm{bad}}(y_i - x_i) \tag{4}$$

where $p(\text{good})$ and $p(\text{bad})$ are the prior probabilities that a measurement is good and an outlier, respectively. I can then compute a posterior probability that the point is an outlier. It is

$$p_{\text{post}}(\text{bad}) = \frac{p_{\text{prior}}(\text{bad})p_{\text{bad}}(y_i - x_i)}{p_{\text{prior}}(\text{bad})p_{\text{bad}}(y_i - x_i) + p_{\text{prior}}(\text{good})p_{\text{good}}(y_i - x_i)} \tag{5}$$

$$= \left(1 + \frac{p_{\text{prior}}(\text{good})p_{\text{good}}(y_i - x_i)}{p_{\text{prior}}(\text{bad})p_{\text{bad}}(y_i - x_i)}\right)^{-1}. \tag{6}$$

We'll do a concrete example, taking the probability distribution for outliers to be a Gaussian with width $10\sigma_i$, and a 50/50 prior probability that a point is an outlier. The likelihood would become

$$\mathcal{L}_i = \frac{1}{2}\frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left[-\frac{(x_i - y_i)^2}{2\sigma_i^2}\right] + \frac{1}{2}\frac{1}{\sqrt{200\pi\sigma_i^2}} \exp\left[-\frac{(x_i - y_i)^2}{200\sigma_i^2}\right] \tag{7}$$

and the posterior probability that a point is an outlier would be

$$p_{\text{post}}(\text{bad}) = \left(1 + \frac{p_{\text{good}}(y_i - x_i)}{p_{\text{bad}}(y_i - x_i)}\right)^{-1} \tag{8}$$

$$= \left(1 + \frac{\sqrt{200\pi\sigma_i^2}}{\sqrt{2\pi\sigma_i^2}} \exp\left[\frac{(x_i - y_i)^2}{200\sigma_i^2} - \frac{(x_i - y_i)^2}{2\sigma_i^2}\right]\right)^{-1} \tag{9}$$

$$= \left(1 + 10\exp\left[-\frac{99}{100}\frac{(x_i - y_i)^2}{2\sigma_i^2}\right]\right)^{-1} \tag{10}$$

$$\tag{11}$$

Suppose that a point has zero residual. Then there is a prior probability of $\frac{1}{2}$ that the point is bad and a posterior probability of $\frac{1}{11}$. Suppose that a point is a 1σ residual. In that case, its posterior probability of being an outlier is 14%. If the point is a 2σ, 3σ, or a 4σ outlier, these posterior probabilities rise to 42%, 90%, and 99.6%, respectively. So this kind of approach smoothly and effectively discounts outliers, and can handle a large fraction of bad data.

Now for the down side: Equation (4) has an addition. If I multiply a bunch of these together, even if I use Gaussian distributions for both the main and the outlier distributions, I don't get to transform a product into a sum with the exponential. I'm stuck with a mixture of products and sums. If there are only a few data points you can still multiply everything out. But taking the log no longer gives me something I can just differentiate and solve with linear algebra routines. You can write down a likelihood but you'll need something else to solve it, like a general optimization technique or, better yet, a sampling technique. We'll introduce Markov Chain Monte Carlo next time; this will be a sampling technique that you can apply to this sort of likelihood.

There is one more approach to the mixture model that deserves special mention: *expectation-maximization.* You can combine the mixture model with iteratively reweighted least squares in the following way. Given the distributions for the outliers and the good data, and a model for the data, you can compute the posterior probability that each point is an outlier. You can then apply weights to each data point y_i in IRLS of

$$w_i = \frac{p(\text{good}|y_i)}{\sigma_i^2}. \tag{12}$$

You would compute these posterior probabilities using something like Equation (6). Once you do that, a linear model with Gaussian errors will still be amenable to a solution by linear algebra techniques, and the general form for the likelihood becomes a bit easier to write out.

Once you start to encounter outliers and have less than absolute confidence in your model for your measurement uncertainties, the first casualties are typically the uncertainties of your fits. These depend sensitively on your measurement errors, and getting these wrong can have very bad effects. You can get around some of this with a mixture model or by throwing out bad data using IRLS. But I'll introduce something more generally applicable here, called *bootstrap error estimation.* This is a bit like magic, and has only recently been accepted by the general statistical community.

The basic idea is the following. You have a set of data, and you have estimated some set of model parameters from it. Maybe you used ordinary χ^2, maybe you used one of the techniques above, maybe you used something a bit different. Now you want to estimate the uncertainties on each of your model parameters. You probably can't trust $\Delta\chi^2 = 1$ for a 1σ contour: this assumes that you got your errors right and that everything is Gaussian. What do you do?

The gold standard for this sort of thing is making a full simulation of your experiment and your data. Then, when you compute a statistic, best-fit parameter, etc. from your data, you can do the same thing on a lot of simulated data sets. This immediately tells you how surprising your result it, and what distribution of fitted parameters you would expect in a universe full of independent realizations of your experiment. End-to-end models are hard, though. This is a major job in high-energy particle physics, because the data are complicated and the stakes are high.

We'll look at two simpler approaches that you can use as a general rule: the bootstrap, and the jackknife. Conceptually they are very similar, though bootstrap has only gained prominence as computers have become good enough to make it practical and statisticians have proved theorems about its applicability. In both cases the basic idea is that you cannot (or choose not to) make a full simulation of your data. Such a simulation would enable you to recompute any statistic of interest on any number of artificial data sets. If you don't have such a simulation, you can instead *assume that the distribution from which the measurements were drawn is approximated by the observed distribution of measurements.* This isn't foolproof–there could be catastrophic outliers in the parent distribution, but I may not have gotten any of these outliers in my data set. Neither bootstrap nor jackknife will protect you against this possibility. I'm not sure that anything will apart from a detailed understanding of your experiment (see the discussion above). This core assumption gives me at least two ways of computing a distribution of a statistic of interest:

- I can leave out a measurement, leaving out each one in turn, and compute scaled statistics on the remainder. This is called jackknife.
- I can create new samples of measurements by drawing from my actual measurements with replacements, and compute the distributions of statistics on these samples. This is called bootstrap resampling.

Let's look at jackknife first.

The jackknife consists of computing a statistic of interest n times, each time leaving out one measurement. Let's call a measured statistic leaving out the i-th data point x_i. The jackknife estimate of the variance on the actual measured value of x is then given by

$$\mathrm{Var}(x) \approx \frac{n-1}{n} \sum_{i=1}^{n} (x_i - \langle x_i \rangle)^2 \tag{13}$$

Using this formula, with the $(n-1)/n$ correction, is the same formula that you use to estimate the variance of a normally distributed quantity from a series of measurements. This formula is derived assuming that the variance on a measurement scales with $1/n$ for large n.

The bootstrap is simpler, so simple that it might feel like cheating. At some level, we drop any assumptions that we know the distribution of errors, but instead assume that the distribution of measurement points is approximated by the observed distribution of points. I can then try to come up with another distribution of observed points and recompute my best-fit parameters. If I assume that my actual measurement points are the best indication I have of their true distribution, then the best thing to do is to draw new measurement samples from this distribution. In other words, with n data points, I make new synthetic data sets of n points each by drawing from my actual data points with replacement. Intuitively, this measures the effect of leaving out particular data or weighting some data more than others. The technique is known as *bootstrap resampling*. I generate many synthetic data sets and compute my best-fit parameters on each data set separately. I can then compute the distribution of best-fit values explicitly without assuming the likelihood to be locally Gaussian, or without even assuming that I have an especially good model for the likelihood.

In pseudo-code, bootstrap resampling looks like this, assuming that my data points are `y` and that the inverse variance of my data (the inverse of my measurement error squared) is `ivar`:

```
for i in range(n_bootstrap):
    i_sample = np.random.randint(0, len(y), len(y))
    y_sample = y[i_sample]
    ivar_sample = ivar[i_sample]
    model[i] = fitmodel(y_sample, ivar_sample)
```

This gives you a bunch of fitted models. You can then compute means, medians, and variances on the sets of models. It's vastly easier than trying to understand exactly how uncertainties in your error distributions propagate to your model confidence intervals, but it means running your model many (probably thousands) of times.