Lectures 14: Bootstrap II.

error propagation for nonlinear functions of fit parameters

with material from
The University of Texas at Austin, CS 395T, Spring 2010, Prof. William H. Press
A CASE STUDY: CHANGE-POINT DETECTION

The task of change-point detection is of major importance in a number of scientific disciplines, ranging from engineering and sociology to economics and environmental studies.

The aim of the change-point identification task is to detect partitions in a sequence of observations, in order for the data in each block to be statistically “similar,” in other words, to be distributed according to a common probability distribution.

Figure 1 shows the number of deadly accidents per year in the coal mines in England spanning the years 1851-1962. Looking at the graph, it is readily observed that the “front” part of the graph looks different from its “back” end, with a change around 1890-1900. As a matter of fact, in 1890, new health and safety regulations were introduced, following pressure from the coal miners’ unions. We will use the poisson distribution.

Coal mining data

x=[4 5 4 1 0 4 3 4 0 6 3 3 4 0 2 6 3 5 4 5 3 1 4 4 1 5 5 3 4 2 5 2 2 ...
3 4 2 1 3 2 2 1 1 1 1 3 0 0 1 0 1 1 0 0 3 1 0 3 2 2 0 1 1 1 0 1 0 1 0 ...
0 0 2 1 0 0 0 1 1 0 2 3 3 1 1 2 1 1 1 2 4 2 0 0 0 1 4 0 0 0 1 0 0 0 ...
0 0 1 0 0 1 0 1]
the $n_0$ probably distribution plot and $\lambda_1$, $\lambda_2$ from Gibbs sampling:

Now back to work so that we can do that!
Fitting is usually presented in frequentist, MLE language. But one can equally well think of it as Bayesian:

\[
P(b | \{y_i\}) \propto P(\{y_i\} | b) P(b) \\
\propto \prod_i \exp \left[ -\frac{1}{2} \left( \frac{y_i - y(x_i | b)}{\sigma_i} \right)^2 \right] P(b) \\
\propto \exp \left[ -\frac{1}{2} \sum_i \left( \frac{y_i - y(x_i | b)}{\sigma_i} \right)^2 \right] P(b) \\
\propto \exp[-\frac{1}{2} \chi^2(b)] P(b)
\]

Now the idea is: Find (somehow!) the parameter value \( b_0 \) that minimizes \( \chi^2 \).

For linear models, you can solve linear “normal equations” or, better, use Singular Value Decomposition. See NR3 section 15.4.

In the general nonlinear case, you have a general minimization problem, for which there are various algorithms, none perfect.

Those parameters are the MLE. (So it is Bayes with uniform prior.)
How accurately are the fitted parameters determined?

As Bayesians, we would instead say, what is their posterior distribution?

Taylor series:

\[-\frac{1}{2} \chi^2(b) \approx -\frac{1}{2} \chi^2_{\text{min}} - \frac{1}{2} (b - b_0)^T \left[ \frac{1}{2} \frac{\partial^2 \chi^2}{\partial b \partial b} \right] (b - b_0)\]

So, while exploring the \( \chi^2 \) surface to find its minimum, we must also calculate the Hessian (2nd derivative) matrix at the minimum.

Then

\[ P(b|\{y_i\}) \propto \exp \left[ -\frac{1}{2} (b - b_0)^T \Sigma_b^{-1} (b - b_0) \right] P(b) \]

with

\[ \Sigma_b = \left[ \frac{1}{2} \frac{\partial^2 \chi^2}{\partial b \partial b} \right]^{-1} \]

covariance (or “standard error”) matrix of the fitted parameters

Notice that if (i) the Taylor series converges rapidly and (ii) the prior is uniform, then the posterior distribution of the \( b \)’s is multivariate Normal, a very useful CLT-ish result!
multivariate normal distribution

Question: What is the generalization of

$$\chi^2 = \sum_i \left( \frac{x_i - \mu_i}{\sigma_i} \right)^2,$$

$$x_i \sim \mathcal{N}(\mu_i, \sigma_i)$$

to the case where the $x_i$'s are normal, but not independent?
I.e., $\mathbf{x}$ comes from a multivariate Normal distribution?

$$N(\mathbf{x}|\mathbf{\mu}, \Sigma) = \frac{1}{(2\pi)^{M/2} \det(\Sigma)^{1/2}} \exp\left[ -\frac{1}{2} (\mathbf{x} - \mathbf{\mu})^T \Sigma^{-1} (\mathbf{x} - \mathbf{\mu}) \right]$$

The mean and covariance of r.v.'s from this distribution are*

$$\mathbf{\mu} = \langle \mathbf{x} \rangle \quad \Sigma = \langle (\mathbf{x} - \mathbf{\mu})(\mathbf{x} - \mathbf{\mu})^T \rangle$$

In the one-dimensional case $\sigma$ is the standard deviation, which can be visualized as “error bars” around the mean.

In more than one dimension $\Sigma$ can be visualized as an error ellipsoid around the mean in a similar way.

$$1 = (\mathbf{x} - \mathbf{\mu})^T \Sigma^{-1} (\mathbf{x} - \mathbf{\mu})$$
Multivariate Normal Distributions

Generalizes Normal (Gaussian) to M-dimensions
Like 1-d Gaussian, completely defined by its mean and (co-)variance
Mean is a M-vector, covariance is a M x M matrix

\[
N(x|\mu, \Sigma) = \frac{1}{(2\pi)^{M/2} \det(\Sigma)^{1/2}} \exp\left[-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu)\right]
\]

The mean and covariance of r.v.’s from this distribution are*

\[
\mu = \langle x \rangle \quad \Sigma = \langle (x - \mu)(x - \mu)^T \rangle
\]

In the one-dimensional case \( \sigma \) is the standard deviation, which can be visualized as “error bars” around the mean.

In more than one dimension \( \Sigma \) can be visualized as an error ellipsoid around the mean in a similar way.

\[
1 = (x - \mu)^T \Sigma^{-1} (x - \mu)
\]
confident intervals

Monte Carlo to generate the parameters:
confidence intervals

Monte Carlo to generate the parameters:

\[ a_{(i)1}^{(s)} - a_{(0)1} \]

68% confidence interval on \( a_0 \)

68% confidence region on \( a_0 \) and \( a_1 \) jointly

68% confidence interval on \( a_1 \)

\[ a_{(i)0}^{(s)} - a_{(0)0} \]

bias
new question: error propagation for arbitrary function of parameters

What is the uncertainty in quantities other than the fitted coefficients:

Method 1: Linearized propagation of errors

\( b_0 \) is the MLE parameters estimate

\( b_1 \equiv b - b_0 \) is the RV as the parameters fluctuate

\[
f \equiv f(b) = f(b_0) + \nabla f b_1 + \cdots
\]

\[
\langle f \rangle \approx \langle f(b_0) \rangle + \nabla f \langle b_1 \rangle = f(b_0)
\]

\[
\langle f^2 \rangle - \langle f \rangle^2 \approx 2f(b_0)(\nabla f \langle b_1 \rangle) + \langle (\nabla f b_1)^2 \rangle
\]

\[
= \nabla f \langle b_1 b_1^T \rangle \nabla f^T
\]

\[
= \nabla f \Sigma \nabla f^T
\]
new question: error propagation for arbitrary function of parameters

In our example, if we are interested in the area of the “hump”,

\[
\begin{array}{cccccc}
\text{bfit} &=& 1.1235 & 1.5210 & 0.6582 & 3.2654 & 1.4832 \\
\text{covar} &=& 0.1349 & 0.2224 & 0.0068 & -0.0309 & 0.0135 \\
& & 0.2224 & 0.6918 & 0.0052 & -0.1598 & 0.1585 \\
& & 0.0068 & 0.0052 & 0.0049 & 0.0016 & -0.0094 \\
& & -0.0309 & -0.1598 & 0.0016 & 0.0746 & -0.0444 \\
& & 0.0135 & 0.1585 & -0.0094 & -0.0444 & 0.0948
\end{array}
\]

\[ f = b_3 b_5 \]

\[ \nabla f = (0, 0, b_5, 0, b_3) \]

\[ \nabla f \Sigma \nabla f^T = b_5^2 \Sigma_{33} + 2b_3b_5 \Sigma_{35} + b_3^2 \Sigma_{55} = 0.0336 \]

\[ \sqrt{0.0336} = 0.18 \]

so \[ b_3 b_5 = 0.98 \pm 0.18 \]

the one standard deviation (1-\(\sigma\)) error bar

A function of normals is not normal
Sampling the posterior histogram

Method 2: Sample from the posterior distribution

1. Generate a large number of (vector) $b$'s

   \[ b \sim \text{MVNormal}(b_0, \Sigma_b) \]

2. Compute your $f(b)$ separately for each $b$

3. Histogram

Note again that $b$ is typically (close to) m.v. normal because of the CLT, but your (nonlinear) $f$ may not, in general, be anything even close to normal!
Sampling the posterior histogram

Our example:

```matlab
bees = mvnrnd(bfit,covar,10000);
humps = bees(:,3).*bees(:,5);
hist(humps,30);
std(humps)
```

`std = 0.1833`

Does it matter that I use the full covar, not just the 2x2 piece for parameters 3 and 5?
comparison of linear propagation and posterior sampling:

Compare linear propagation of errors to sampling the posterior

- Note that even with lots of data, so that the distribution of the b’s really \( \rightarrow \) multivariate normal, a derived quantity might be very non-Normal.
  - In this case, sampling the posterior is a good idea!
- For example, the ratio of two normals of zero mean is Cauchy
  - which is very non-Normal!
- So, sampling the posterior is a more powerful method than linear propagation of errors.
  - even when optimistically (or in ignorance) assuming multivariate Gaussian for the fitted parameters
- In fact, sampling the posterior distribution of large Bayesian models whose parameters are not at all Gaussian is, under the name MCMC, the most powerful technique in modern computational statistics.
Method 3: Bootstrap resampling of the data

- We applied some end-to-end process to a data set and got a number out.
- The data set was drawn from a population of repetitions of the identical experiment
  - which we don’t get to see, unfortunately
  - we see only a sample of the population
- We’d like to draw new data sets from the population, reapply the process, and see the distribution of answers
  - this would tell us how accurate the original answer, on average, was
  - but we can’t: we don’t have access to the population
- However, the data set itself is an estimate of the population pdf!
  - in fact, it’s the only estimate we’ve got!
- So we draw from the data set – with replacement – many “fake” data sets of equal size, and carry out the proposed program
  - does this sound crazy? for a long time many people thought so!
  - Bootstrap theorem [glossing over technical assumptions]: The distribution of any resampled quantity around its full-data-set value estimates (naively: “asymptotically has the same histogram as”) the distribution of the data set value around the population value.
Let’s try a simple example where we can see the “hidden” side of things, too.

**Visible side (sample):**

**Hidden side (population):**

These happen to be drawn from a Gamma distribution.

**Statistic we are interested in happens to be (it could be anything):**

\[
\frac{\text{mean of distribution}}{\text{median of distribution}}
\]

```python
sammedian = median(sample)
sammean = mean(sample)
samstatistic = sammean/sammedian
sammedian =
  2.6505
sammean =
  2.9112
samstatistic =
  1.0984
```

```python
themedian = median(bigsample)
themean = mean(bigsample)
thestatistic = themean/themedian
themedian =
  2.6730
themean =
  2.9997
thestatistics =
  1.1222
```

How accurate is this?
Gamma distribution:

\[ x \sim \text{Gamma}(\alpha, \beta), \quad \alpha > 0, \beta > 0 \]

\[ p(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}, \quad x > 0 \]

Mean\{\text{Gamma}(\alpha, \beta)\} = \frac{\alpha}{\beta}

Var\{\text{Gamma}(\alpha, \beta)\} = \frac{\alpha}{\beta^2}

When \( \alpha \geq 1 \) there is a single mode at \( x = (\alpha - 1)/\beta \)
To estimate the accuracy of our statistic, we bootstrap

new sample of integers in 1:ndata, with replacement

```
ndata = 100;
nboot = 100000;
vals = zeros(nboot,1);
for j=1:nboot,
    choose = randsample(ndata,ndata,true);
    vals(j) = mean(sample(choose))
        /median(sample(choose));
end
hist(vals,100)
```

```
ndata = 100;
nboot = 100000;
vals = zeros(nboot,1);
for j=1:nboot,
    sam = randg(3,[ndata 1]);
    vals(j) = mean(sam)/median(sam);
end
hist(vals,100)
```

Things to notice:
The mean of resamplings does not improve the original estimate! (Same data!)
The distribution around the mean is not identical to that of the population. But it is close and would become identical asymptotically for large ndata (not nboot!).
bootstrap sampling

ndata = 20;
nboot = 1000;
vals = zeros(nboot,1);
ynmodel = @(x,b) b(1)*exp(-b(2)*x)+b(3)*exp(-(1/2)*((x-b(4))/b(5)).^2);
for j=1:nboot,
    samp = randsample(ndata,ndata,true);  % new sample of integers in 1:ndata, with replacement
    xx = x(samp);
    yy = y(samp);
    ssig = sig(samp);
    chisqfun = @(b) sum(((ynmodel(xx,b)-yy)./ssig).^2);
    bguess = [1 2.7 3.14 1.5];
    options = optimset('MaxFunEvals',10000,'MaxIter',10000,'TolFun',0.001);
    [b fval flag] = fminsearch(chisqfun,bguess,options);
    if (flag == 1), vals(j) = b(3)*b(5);
    else vals(j) = 100; end
end
hist(vals(vals < 2),30);
std(vals(vals < 2))

0.2924

So we get the peak around 1, as before, but a much broader distribution.
Can you guess what the extreme bootstrap cases look like, compared to the full data?

**$b_3b_5 \approx 0.3$**

**$b_3b_5 \approx 5$**

frequentist is concerned about error estimate
We previously compared bootstrap-from-sample to bootstrap-from-population. More relevant, let's compare bootstrap-from-sample to sample-the-posterior:

- We could increase number of samples of posterior, and of bootstrap, to make both curves very smooth.
  - the histograms would not converge to each other!
- We could increase the size of the underlying data sample
  - from 20 (x,y) values to infinity (x,y) values
  - the histograms **would** converge to each other (modulo technical assumptions)
- For finite size samples, each technique is a valid answer to a different question
  - Frequentist: Imagining repetitions of the experiment, what would be the range of values obtained?
    - And, conservatively, I shouldn't expect my experiment to be better than that, should I?
  - Bayesian: For exactly the data that I see, what is the probability distribution of the parameters?
    - Because maybe I got lucky and my data set really nails the parameters!
bootstrap sampling

Note that sampling the posterior "honors" the stated measurement errors. Bootstrap doesn't. That can be good!

Suppose (very toy example) the "statistic" is

\[ s = x_1 + x_2 \]

then the posterior probability is

\[ P(s) \propto \exp \left[ -\frac{1}{2} \left( \frac{s - x_1 - x_2}{\sigma_1^2 + \sigma_2^2} \right)^2 \right] \]

Note that this depends on the \( \sigma \)'s!

The bootstrap (here noticeably discrete) doesn't depend on the \( \sigma \)'s. In some sense it estimates them, too.

So, if the errors were badly underestimated, sampling the posterior would give too small an uncertainty, while bootstrap would still give a valid estimate.

If the errors are right, both estimates are valid. Notice that the model need not be correct. Both procedures give estimates of the statistical uncertainty of parameters of even a wrong (badly fitting) model. But for a wrong model, your interpretation of the parameters may not mean anything!
bootstrap sampling

Compare and contrast bootstrap resampling and sampling from the posterior. Both have the same goal: Estimate the accuracy of fitted parameters.

- **Bootstrap** is frequentist in outlook
  - draw from “the population”
  - even if we have only an estimate of it (the data set)
- **Easy to code but computationally intensive**
  - great for getting your bearings
  - must repeat your basic fitting calculation over all the data 100 or 1000 times
- **Applies to both model fitting and descriptive statistics**
- **Fails completely for some statistics**
  - e.g. (extreme example) “harmonic mean of distance between consecutive points”
  - how can you be sure that your statistic is OK (without proving theorems)?
- **Doesn’t generalize much**
  - take it or leave it!
- **It is not always obvious what you should resample over**
  - things that are independent draws from a population

- **Sampling from the posterior** is Bayesian in outlook
  - there is only one data set and it is never varied
  - what varies from sample to sample is the goodness of fit of the parameters
  - we don’t just sit on the (frequentist’s) MLE, we explore around
- **In general harder to implement**
  - we haven’t learned how yet, except in the simple case of an assumed multivariate normal posterior
  - will come back to this later, when we do Markov Chain Monte Carlo (MCMC)
  - may or may not be computationally intensive (depending on whether there are shortcuts possible in computing the posterior)
- **Rich set of variations and generalizations are possible**