Parameter estimation tutorial 1

S. L. Bridle 29/09/2006

Overview

This tutorial generates a mock data set and tries to reconstruct the parameters used to generate the mock data. The mock data set is drawn from a straight line with offset c and slope m. Therefore the data consists of n pairs of x,y values. The aim is to recover the parameters c and m and find how well they are constrained (value and error bars).

This will involve:

- Random number generation
- Concept of Gaussian errors on data
- The likelihood function for a single data point
- Combining likelihoods for many data points
- Conditional likelihoods
- One dimensional probability distributions
- Deriving error bars on parameters
- Two dimensional probability distributions
- Marginalisation

Although this tutorial can be carried out in any programming language, matlab is used for specific examples.

Task 1: Simulation of data

- Choose some x values

  e.g. >> n=10

  >> x = 10*rand(n,1)

  Check: >> hist(x)
- Choose your straight line parameters

  e.g. >> m_true=0.2; c_true=2;

  Check: >> m_true
- Generate noise-free y values

  >> y_nonoise=m_true*x +c_true

  Check: >> plot(x,y_nonoise,'bx')
- Add Gaussian noise

  e.g. >> sigma=0.2*ones(n,1)

  >> y=m_true*x + c_true + sigma.*randn(n,1)

  Check: >> plot(x,y,'bx')
- Make a nice plot

  >> clf; % clear the plot

  >> errorbar(x,y,sigma,'ro')

  >> hold on; % don't overwrite the above

  >> plot(x,y_nonoise,'b-')

  >> xlabel('x');

  >> ylabel('y');

  >> legend('Simulated data','Theory line')

Task 2:
Calculate a trial likelihood

- Pick some trial straight line parameters
  >> m_trial=0.4, c_trial=1
- Make predictions for this model
  >> y_trial = m_trial*x+c_trial
  Check: >> plot(x,y_trial,'gx')
- Calculate χ^2
  >> chisq=sum((y-y_trial).^2./sigma.^2)
- Calculate the unnormalised likelihood
  >> like = exp(-chisq/2)
- Deeper understanding: Why is the likelihood of this form? Hint: this assumes that the errors on the data are Gaussian.
  - Why is it called the "unnormalised" likelihood?
- Repeat the above but using the true input values
  >> m_trial=m_true; c_trial=c_true
- Make a loop around the whole procedure and repeat
  Put the below into a file e.g. called toy_chisq.m
  ```
  >> clear; % get rid of any stored variables
  >> nreal=5
  >> for ireal=1:nreal
  >>   ... all the above stuff...
  >>   chisq_store(ireal)=chisq
  >> end
  >> hist(chisq_store)
  ```
  If the above doesn't go crazy, can now increase nreal to 100 or even 1000.
- Where does the histogram peak?
- What is its width?
- Can you relate these to the number of data points?
  Repeat the above but with a different number of data points to test your theory.
- Could practice using the matlab debugger here
  Open toy_chisq.m in the matlab editor (if it isn't already open) and click in the margin to produce a red dot (breakpoint).
  Press f5 to save and execute the script. It should stop at the breakpoint. You can now type variable names at the prompt and see their current values.
  Press f5 again to continue running the program, until it reaches the breakpoint again.
  Get rid of the breakpoint by clicking on the red dot.
  Press F5 again to let it run without interruptions (now the breakpoint has gone).

Task 3:
Calculate 1d conditional probabilities

- Vary one of your trial parameters and calculate the likelihood as a function of that parameter

  Put the below e.g. in toy_1d_like.m

  ```
  >> clear
  >> m_trial_vals=0:0.05:0.5
  >> c_trial = 1
  >> for im=1:length(m_trial_vals)
  >>   m_trial=m_trial_vals(im)
  >> ... repeat stuff from prev page...
  >>   chisq_vals(im) = chisq
  >> end
  ```

  Check: `>> plot(m_trial_vals,chisq_vals,'bx')`

  ```
  >> chisq_vals_norm=chisq_vals –
     min(chisq_vals)
  >> like_vals = exp(-chisq_vals_norm/2)
  >> plot(m_trial_vals,like_vals)
  ```

- Find the most likely m value
  – Ask Alex...
- Find the 68 per cent limits on m
  – Ask Alex...

Task 4:
Calculate the full 2d probability distribution

- Vary both m and c
- Plot the 2d chisq map
  - Use >> chisq_vals(im,ic)=chisq;
  - Check >> imagesc(m_trial_vals,c_trial_vals,chisq_vals)
  - >> contour(m_trial_vals,c_trial_vals,chisq_vals)
  - Use a different number of values in m_trial_vals cf c_trial_vals to make sure your plot is coming out right (e.g. only 3 different m_trial_vals but 10 different c_trial_vals)
  - Play with the ranges on the parameters
- Use a Gaussian approximation to calculate 68 and 95 percent confidence contours
  - Ask Alex...

Task 5:
Calculate marginalised error bars

- Ask Lisa...