In-Class Computing Project

Statistical Computing & Machine Learning

Likelihood calculations

A likelihood is a probability of a set of observations given a model. In the simplest terms, a probability is a number between zero and one. In practice, one works with the logarithm of the likelihood, which will be a number between 0 and $-\infty$. Of course, at the point where the likelihood has a maximum, the log-likelihood will also have a maximum. The reason to prefer log-likelihoods is that likelihoods can be very, very small, beyond the capabilities of numerical calculations in computer hardware. The logarithm of the likelihood, however, is easily used in numerical calculations.

In this activity, you will perform some likelihood calculations on data simulated from a known source. Then you will find the maximum log-likelihood and check to see if the parameters are a good match to the simulation.

Exponential models

The exponential distribution has one parameter, `rate`.

1. Recall that our objective here is to make an estimate of this parameter from data. It's convenient to do this with a simulation. This way, you'll know what the estimated rate *should be* and you'll be able to see how well the estimator you build is performing.

   Create an object `vals` containing 10 random values selected from an exponential distribution with rate 1/100. You can use `rexp()` for this. Check your values to make sure they are plausible. If an event happens at a rate of 1/100, for instance the so-called 100-year storm, you expect the interval between events to be something like 100. But because the events are random, the intervals will not usually be exactly 100 years.

2. Create an object `test200` that gives the log-likelihood of `vals` for a rate of 1/200. You can use `dexp()` for this, along with `sum()` and `log()`. The `dexp()` function will calculate the likelihood of each of the 10 values independently in `vals`. Ordinarily, you would multiply them together to find the likelihood of the 10 values taken as a set. For log-likelihood, take the log of the output of `dexp()` — producing a 10 number result. Then *sum* over those numbers to get the log-likelihood.

3. Write a function `LL_exp(rate)` that computes the log-likelihood

of `vals` for the given `rate`. Test it out by comparing the result to what you got in (2).

4. Create an object `rates` that is a vector of many rates between 1/50 and 1/200.

5. You are going to compute the log-likelihood of `vals` for each component of `rates`.[1] The following command will do the job:

```
results <- sapply(rates, LL_exp)
```

6. Plot out `results` against `1/rates`. By eye, find the maximum value. Actually, it's not so much the log-likelihood value we are interested in but the value of `1/rates` at which that maximum value occurs. This is called the *argmax*. Note that you will want your vector `rate` to be fine enough that you get a graph that looks continuous.

[1] Those of you who have experience writing computer programs will think of using a loop. R has functions that perform the task of looping and accumulating answers for you.

Automating the search

The R function `optimize()` will do a search over an interval for the argmax or argmin of a function. You're going to find the argmax of your `LL_exp(rate)` from the previous activity function in this way. To use `optimize()` you specify the function that for which you're searching the argmax, an interval `lower=` to `upper=` and set the argument `maximum = TRUE`. (If you don't specify `maximum = TRUE`, `optimize()` will search for an argmin.)

7. Create an object `exp_results` that contains the output of `optimize()`. Check it to make sure that your answer is consistent with that in Task 1.

```
exp_results <- optimize(LL_exp, lower = 1/200, upper = 1/50, maximum = TRUE)
```