

# Computer Lab 6

## Computational Statistics

Linköpings Universitet, IDA, Statistik

2023 XII 8

---

Kurskod och namn:	732A90 Computational Statistics
Datum:	2023 XII 05—2023 XII 15 (lab session 8 XII 2023, SU12/13/14)
Delmomentsansvarig:	Krzysztof Bartoszek, Bayu Brahmantio, Héctor Rodríguez Déniz
Instruktioner:	<p>This computer laboratory is part of the examination for the Computational Statistics course</p> <p>Create a group report, (that is directly presentable, if you are a presenting group), on the solutions to the lab as a <b>.PDF</b> file.</p> <p>Be concise and do not include unnecessary printouts and figures produced by the software and not required in the assignments.</p> <p><b>All R code should be included as an appendix into your report.</b></p> <p>A typical lab report should 2-4 pages of text plus some amount of figures plus appendix with codes.</p> <p>In the report reference <b>ALL</b> consulted sources and disclose <b>ALL</b> collaborations.</p> <p><b>Copying solutions from others, the Internet and other sources is NOT allowed.</b></p> <p>The report should be handed in via LISAM (or alternatively in case of problems by e-mail), by <b>23:59 22 December 2023</b> at latest.</p> <p>Notice there is a deadline for corrections <b>23:59 31 January 2024</b> and a final deadline of <b>23:59 29 February 2024</b> after which no submissions nor corrections will be considered and you will have to redo the missing labs next year.</p> <p>The seminar for this lab will take place <b>12 December 2023, 10:15, KY27</b>.</p> <p>The report has to be written in English.</p>

---

## Question 1: Genetic algorithm

In this assignment, you will try to solve the  $n$ -queen problem using a genetic algorithm. Given an  $n$  by  $n$  chessboard, the task is to place  $n$  queens on it so that no queen is attacked by any other queen. You can read more about the problem at [https://en.wikipedia.org/wiki/Eight\\_queens\\_puzzle](https://en.wikipedia.org/wiki/Eight_queens_puzzle).

1. An individual in the population is a chessboard with some placement of the  $n$  queens on it. The first task is to code an individual. You are to consider three encodings for this question.
  - (a) A collection (e.g., a list—but the choice of data structure is up to you) of  $n$  pairs denoting the coordinates of each queen, e.g.,  $(5, 6)$  would mean that a queen is standing in row 5 and column 6. We are not using chess notation (in this case e5) as we do not want to limit  $n$  by 26 and also further work will be easier with a numerical notation.
  - (b) On  $n$  numbers, where each number has  $\log_2 n$  binary digits—this number encodes the position of the queen in the given column. Notice that as queens cannot attack each other, in a legal configuration there can be only one queen per column. You can pad your binary representation with 0s if necessary.
  - (c) On  $n$  numbers, where each number is the row number of the queen in each column. Notice that this encoding differs from the previous one by how the row position is stored. Here it is an integer, in item 1b it was represented through its binary representation. This will induce different ways of crossover and mutating the state.

The tasks below, 2–7 are to be repeated for each of the three encodings above.

2. Define the function `crossover()`: for two chessboard layouts it creates a kid by taking columns  $1, \dots, p$  from the first individual and columns  $p + 1, \dots, n$  from the second. Obviously,  $0 < p \leq n/2$ , and  $p \in \mathbb{N}$ . Experiment with different values of  $p$ .
3. Define the function `mutate()` that randomly moves a queen to a new position.
4. Define a fitness function for a given configuration. Experiment with three: binary—is a solution or not; number of queens not attacked;  $\binom{n}{2}$ -number of pairs of queens attacking each other. If needed scale the value of the fitness function to  $[0, 1]$ . Experiment which could be the best one. Try each fitness function for each encoding method. You should not expect the binary fitness function to work well, explain why this is so.
5. Implement a genetic algorithm that takes the choice of encoding, mutation probability, and fitness function as parameters. Your implementation should start with a random initial configuration. Each element of the population should have its fitness calculated. Do not forget to have in your code a limit for the number of iterations (but this limit should not be lower than 100, unless this causes running time issues, which should be clearly presented then), so that your code does not run forever. Count the number of pairs of queens attacking each other. At each iteration :

- (a) Two individuals are randomly sampled from the current population, they are further used as parents (use `sample()`).
  - (b) One individual with the smallest fitness is selected from the current population, this will be the victim (use `order()`).
  - (c) The two sampled parents are to produce a kid by crossover, and this kid should be mutated with probability `mutprob` (use `crossover()`, `mutate()`).
  - (d) The victim is replaced by the kid in the population.
  - (e) Do not forget to update the vector of fitness values of the population.
  - (f) Remember the number of pairs of queens attacking each other at the given iteration.
6. If found return the legal configuration of queens.
  7. Provide a plot of the number of pairs queens attacking each other at each iteration of the algorithm.
  8. Run your code for  $n = 4, 8, 16$  (if  $n = 16$  requires too much computational time take a different  $n \in \{10, 11, 12, 13, 14, 15\}$ , but do not forget that this is not a power of 2 and more care is needed in the second encoding), the different encodings, objective functions, and `mutprob`= 0.1, 0.5, 0.9. Did you find a legal state?
  9. Discuss which encoding and objective function worked best.

## Question 2: EM algorithm

The data file `censoredproc.csv` contains the time after which a certain product fails. Some of these measurements are left-censored (`cens=2`)—i.e., we did not observe the time of failure, only that the product had already failed when checked upon. Status `cens=1` means that the exact time of failure was observed.

1. Plot a histogram of the values. Do it for all of the data, and also when the censored observations are removed. Do the histograms remind of an exponential distribution?
2. Assume that the underlying data comes from an exponential distribution with parameter  $\lambda$ . This means that observed values come from the exponential  $\lambda$  distribution, while censored from a truncated exponential distribution. Write down the likelihood function.
3. **The goal now is to derive an EM algorithm that estimates  $\lambda$ .** Based on the above found likelihood function, derive the EM algorithm for estimating  $\lambda$ . The formula in the  $M$ -step can be differentiated, but the derivative is non-linear in terms of  $\lambda$  so its zero might need to be found numerically.
4. Implement the above in R. Take  $\lambda_0 = 100$  as the starting value for the algorithm and stopping condition if the change in the estimate is less than 0.001. At what  $\hat{\lambda}$  did the EM algorithm stop at? How many iterations were required?
5. Plot the density curve of the  $\exp(\hat{\lambda})$  distribution over your histograms in task 1.
6. Study how good your EM algorithm is compared to usual maximum likelihood estimation with data reduced to only the uncensored observations. To this end we will use a parametric bootstrap. Repeat 1000 (reduce if computational time is too long—but carefully report the running times) times the following procedure:
  - (a) Simulate the same number of data points as in the original data, from the exponential  $\hat{\lambda}$  distribution.
  - (b) Randomly select the same number of points as in the original data for censoring. For each observation for censoring—sample a new time from some distribution on  $[\text{true time}, \infty)$ . Remember that the observation was censored.
  - (c) Estimate  $\lambda$  both by your EM-algorithm, and maximum likelihood based on the uncensored observations.

Compare the distributions of the estimates of  $\lambda$  from the two methods. Plot the histograms, report whether they both seem unbiased, and what is the variance of the estimators.