Examination

Linköping University, Department of Computer and Information Science, Statistics

Course code and name	732A38 Computational Statistics
Date and time	2016-05-09, 14.00-19.00
Assisting teacher	Oleg Sysoev
Allowed aids	"Computational statistics" by Gentle or "Computational Statistics" by Givens &Hoeting
Grades:	A=19-20 points B=16-18 points C=11-15 points D=9-10 points E=7-8 points F=0-6 points

Provide a detailed report that includes plots, conclusions and interpretations. Give motivated answers to the questions. If an answer is not motivated, the points are reduced. Provide all necessary codes in the appendix.

Assignment 1 (10p)

1. Implement a function that generates a random sample with *n* observations from the distribution with the density $p(y) \propto a^y, 0 < y < 1$ by using the inverse CDF method. In the formula for $p(y)$, assume that a is some constant, $a \in (0,1)$. Explain how your implementation was obtained step by step. Generate a sample with $a = 0.1$ and $n = 1000$ and plot a histogram of the sample. Does it look like $p(y)$? **(4p)**
    a. **Hint**: Recall $a^y = e^{y \log a}$
2. Implement a Gibbs sampler that generates a random sample with *n* observations from the distribution with the density $p(x, y) \propto x^y(1-x)^3, 0 < x < 1, 0 < y < 1$. **Hint**: in the sampler you may use the generator obtained in step 1, check also the document containing the list of standard univariate distributions (distributions.pdf). Generate trace plots, comment on burn-in period, mixing and convergence. **(4p)**

a. In case you did not succeed with step 1, you may instead implement a generator that uses acceptance/rejection method with multivariate uniform distribution as a majorizing density, but the points will be reduced.

3. Estimate the expected value of $g(x, y) = x^2 + y^2$ by using the distribution simulated in step 2. Compute also the variance of this expected value. **(2p)**

Assignment 2 (10p)

Data file "**X.csv**" contains a sample from chi-square distribution with an unknown amount of degrees of freedom (df).

1. Implement a function **fitness** that depends on the sample X and parameter *Y* and returns minus one divided by the log-likelihood of the sample, assuming that X is chi-square distributed with degrees of freedom *df=Y*. Implement also functions
   - **Crossover** that depends on X and Y and returns a list with $floor(\frac{2X+Y}{3})$ and $floor(\frac{X+2Y}{3})$
   - **Mutation** that depends on X and returns (X*2559+107)%%311 **(2p)**

2. In order to find the unknown parameter *df*, one could optimize *fitness* by using common methods for unconstrained optimization. However, since it is known from the application that *df* is a discrete value, you will need to apply genetic algorithm by implementing function *genetic(X, mutprob)* as follows:
   - Use initial sample Y=1,2,3,…50
   - Repeat the following *100* times
     i. Compute fitness for each element in the sample and select a subsample with 20 elements by using random sampling with replacement where probability of selection is proportional to the fitness. (**Hint**: use sample())
     ii. Creating a new population by repeating the following step 25 times:
        1. Selecting two parents randomly from the subsample
        2. Creating their children by crossover.
        3. Selecting candidate 1 as either parent 1 or kid 1, choose the one with higher fitness
        4. Selecting candidate 2 as either parent 2 or kid 2, choose the one with higher fitness
        5. Mutating candidate 2 with probability *mutprob*
        6. Adding candidate 1 and candidate 2 to the new population.
     iii. Replacing the old population with the new population
   - Return the final population, best fitness of the population and parameter value (degrees of freedom) corresponding to the best fitness.

     Test your function three times for each of the following settings (X here is the original data):
     i. X=X, Mutprob=0.01
     ii. X=X, Mutprob=2
     iii. X=X, Mutprob=0.9

and explain how the mutation probability affects the final sample and the optimization quality and what the reason of this phenomenon is. **(4p)**

3. Use non-parametric bootstrap with B=100 iterations and function **genetic** with mutprob=0.2 to test whether the degrees of freedom parameter is greater than 200. What is the p-value of this test? **(4p)**
    a. In case you didn't succeed with step 2, you may estimate *df* by applying function optimize() to the fitness while doing the bootstrap, but the points will be reduced.