Stochastic optimization algorithms 2021
Home problems, set 1

General instructions. READ CAREFULLY!

Problem set 1 consists of three parts. Problems 1.1 and 1.3 are mandatory, Problem 1.2 is voluntary (but check the requirements for the various grades on the web page). After solving the problems, collect your answers, your programs, *and* your report (see below) in *one* .zip (or .7z) file which, when opened, generates one main folder containing your report and additional subfolders for each problem (e.g. Problems 1.1 and 1.3) in the assignment. In problems that do not involve computer programming (e.g. Problem 1.2, in this case), no folder is needed.

You should provide a single report (for Problems 1.1-1.3, i.e. the whole set) in the form of a PDF file (Note: *Only* this format is accepted). The required sections of the report are specified in the individual problems below. In the case of analytical problems, make sure to include *all the relevant steps of the calculation* in your report, so that the calculations can be followed. Providing only the answer is *not* sufficient. Whenever possible, use symbolical calculations as far as possible, and introduce numerical values only when needed. You should write the report on a computer, preferably using LaTeX (see `www.miktex.org`). Scanned, handwritten pages are *not* allowed.

In *all* problems requiring programming, use Matlab. The *complete* Matlab program for the problem in question (i.e. all source files) must be handed in, collected in the folder(s) for the problem in question. Do not include unnecessary files, e.g. unused Matlab files. It should *not* be necessary to edit the programs, move files etc. Furthermore, when writing Matlab programs, you should make sure to follow the coding standard (available on the web page). You may, however, hardcode *parameters* (in your .m-files), as is indeed done in some of the skeleton files from which you should start. Programs that do not function, or require editing to function, or deviate significantly from the coding standard, or otherwise do not follow the requirements given above, will be returned, resulting in a deduction of points.

The maximum number of points for problem set 1 is 10. Incorrect problems will be returned for correction *only* in cases where the mandatory requirements have not been met, so please make sure to check your solutions and programs carefully before submitting the (single) compressed file (.zip or .7z) in Canvas.

You may, of course, discuss the problems with other students. However, each student *must* hand in his or her *own* solution. Note that a plagiarism check will be carried out, in which both your report and your code are checked against reports and code from other students (both from this year and earlier years). In obvious cases of plagiarism, points will be deducted from all students involved.

NOTE: Don't forget to write your name *and* civic registration number on the front page of the report! Make sure to keep copies of the files that you hand in! Good luck!

(Strict) deadline: 20210921, 23.59.59

Problem 1.1, 3p, Penalty method (Mandatory)

In this problem, we shall use the penalty method (see pp. 30-33 in the course book) to find the minimum of the function

$$f(x_1, x_2) = (x_1 - 1)^2 + 2(x_2 - 2)^2, \tag{1}$$

subject to the constraint

$$g(x_1, x_2) = x_1^2 + x_2^2 - 1 \leq 0. \tag{2}$$

You should now do the following steps:

1. Define (and specify clearly, in your report, as a function of x_1, x_2, and μ) the function $f_{\mathrm{p}}(\mathbf{x}; \mu)$, consisting of the sum of $f(x_1, x_2)$ and the penalty term.

2. Next, compute (analytically) the gradient $\nabla f_{\mathrm{p}}(\mathbf{x}; \mu)$, and include it in your report. Make sure to include both the case where the constraints are fulfilled and the case where they are not.

3. Find (analytically) the unconstrained minimum (i.e. for $\mu = 0$) of the function, and include it in your report. This point will be used as the starting point for gradient descent.

4. Write a Matlab program for solving the unconstrained problem of finding the minimum of $f_{\mathrm{p}}(\mathbf{x}; \mu)$ using the method of gradient descent. You *should* make use of the skeleton files available on Canvas, namely

   (a) The main file `RunPenaltyMethod.m` (that calls the other functions, generates and prints output etc. etc.). Here, you should set a suitable starting point (see above) and you may modify parameters (e.g. the values of μ or the tolerance parameter), but may otherwise not make changes.

   (b) `RunGradientDescent` where you should implement a function that takes the starting point $\mathbf{x}_0$ (as a vector with two elements), the value of μ, the step length (for gradient descent) η, and a threshold T (see below) as input, and carries out gradient descent until the (L2) norm of the gradient, $\|\nabla f_{\mathrm{p}}(\mathbf{x}; \mu)\|$, drops below the threshold T. Use the unconstrained minimum as the starting point; see above.

   (c) `ComputeGradient.m` where you should implement a function that takes as input the vector x (a vector with two elements) and μ, and returns (for any point $x \in \mathbf{R}^2$) the gradient of $f_{\mathrm{p}}(\mathbf{x}; \mu)$ (a vector with two elements). Note: You should hardcode the gradient in this method, i.e. you do not need to write a general method for finding the gradient. However, your method should make use of the analytical gradient, computed in Step 2 above. You should *not* use a numerical approximation of the gradient.

5. Run the program for a suitable sequence of μ values (see `RunPenaltyMethod.m`). Select a suitable (small) value for the step length η, and specify it clearly, along with the sequence of μ values, in your report. Example of suitable parameter values: $\eta = 0.0001$, $T = 10^{-6}$, sequence of μ values: 1, 10, 100, 1000. As output, the program gives the components of the vector x for the different μ values. You should include the table in your report; see below. Specify the values of x_1^* and x_2^* with 4 decimal precision. Do *not* just print the raw Matlab output (with many decimals, for example) in your report! You should also check that your results are reasonable, i.e. that the sequence of points appears to be convergent, for example by plotting the values of x_1 and x_2 as functions of μ.

What to hand in You should hand in your Matlab program, consisting of (exactly) the three files described above (see Step 4 in the list). In your report, you should include a section named *Problem 1.1*, containing the calculations and results for Steps 1 - 3 in the list above, as well as the table with the output from your program, along with a brief discussion regarding the convergence of the point sequences, as described under Step 5 above. Make sure (where applicable) to include any relevant intermediate steps, not just writing down the final result! Omitting intermediate steps may result in the solution being returned, with resulting point deductions; see below.

Maximum number of points for this problem: 3p.
Maximum number of points if your solutions must be returned for correction: 1p

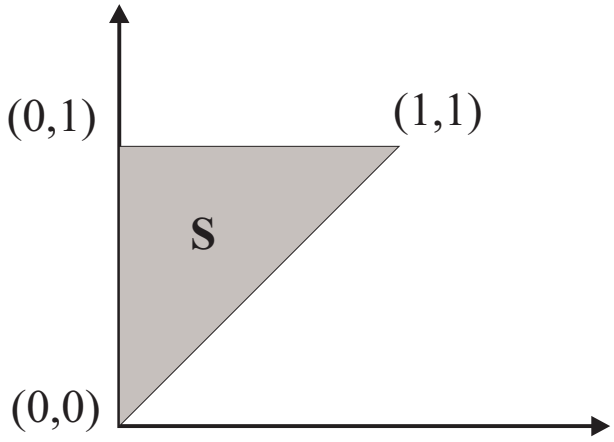

Figure 1: The set S used in Problem 1.2a.

Problem 1.2, 3p, Constrained optimization (Voluntary)

a) (2p) Use the analytical method described on pp. 29-30 in the course book to determine the global minimum $(x_1^*, x_2^*)^{\mathrm{T}}$ (as well as the corresponding function value) of the function

$$f(x_1, x_2) = 4x_1^2 - x_1x_2 + 4x_2^2 - 6x_2, \tag{3}$$

on the (closed) set S, shown in the figure. The corners of the triangle are located at $(0, 0), (0, 1)$ and $(1, 1)$.

b) (1p) Use the Lagrange multiplier method described on pp. 25-28 in the course book to determine the minimum $(x_1^*, x_2^*)^{\mathrm{T}}$ (as well as the corresponding function value) of the function $f(x_1, x_2) = 15 + 2x_1 + 3x_2$ subject to the constraint $h(x_1, x_2) = x_1^2 + x_1x_2 + x_2^2 - 21 = 0$.

What to hand in These two problems should be solved analytically (i.e. by hand). Thus, no Matlab program should be handed in. In your report, write a section *Problem1.2*, where you show (and motivate, where necessary) the relevant intermediate steps in the calculations, as well as the final result.

Problem 1.3, 4p, Basic GA program (Mandatory)

a) In this problem, you will implement a genetic algorithm (GA) for finding the minimum of the function

$$g(x_1, x_2) = (1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2 + (2.625 - x_1 + x_1 x_2^3)^2 \tag{4}$$

You must use the skeleton files provided for this problem on the Canvas page. There are two main files, `RunSingle.m` for running the GA once and `RunBatch.m` for making multiple runs and also obtaining some statistics. In `RunSingle.m` you may *only* modify those parameters marked *changes allowed*. The allowed changes in `RunBatch.m` will be described in part (b) below. These two Matlab scripts both access the function `RunFunctionOptimization`, in the file `RunFunctionOptimization.m`, which you may *not* modify. Now do the following:

Using the skeleton files (in which you may *not* change the interfaces) on the Canvas page, and making sure to follow the coding standard, write functions for...

1. ... initializing a population (`InitializePopulation`),
2. ... decoding a (binary) chromosome (`DecodeChromosome`),
3. ... evaluating an individual (`EvaluateIndividual`),
4. ... selecting individuals with tournament selection (`TournamentSelect`),
5. ... carrying out crossover (`Cross`),
6. ... carrying out mutations (`Mutate`).

Versions of these functions have been implemented during the Matlab introduction. However, for this problem you will make (some of) the functions more general, as follows:

`InitializePopulation`: This function should take the population size and the number of genes as input, and should return the entire population as a matrix of binary numbers (i.e. as in the Matlab introduction).

`DecodeChromosome`: This function should take as input (i) a (binary) chromosome, (ii) the number of variables that are to be extracted, and (iii) the maximum varialbe value. Let m denote the chromosome length and n the number of variables, and let $k = m/n$. The first k bits should be used when forming x_1, the next k bits should be used for generating x_2 etc. Each variable should be decoded from the k bits according to Eq. (3.9) in the course book. You may assume that m and n have been chosen such that k is an integer.

`EvaluateIndividual`: This function should take the vector of variables (`x`) as input, and should return the (note!) *fitness value*, which in this case should be taken as the inverse of the function value, i.e. $1/g(x_1, x_2)$. You should hardcode the function in `EvaluateIndividual` (i.e. you should *not* place the function in a separate file) and then also hardcode the computation of the fitness value. Check carefully that you enter the function $g(x_1, x_2)$ correctly!

`TournamentSelect`: This function should take as input (i) the vector of fitness values (from the most recently evaluated population) (ii) the tournament selection parameter and (iii) the tournament size, and should return the index of the selected individual, using tournament selection. Note that the function should also handle cases where the tournament size is different from 2! See the description near the top of p. 50 in the course book.

`Cross`: This function should take two chromosomes as input, carry out single-point crossover, and return a chromosome pair (i.e. as in the Matlab introduction).

`Mutate`: The `Mutate` function should take as input (i) a chromosome and (ii) a mutation probability, and should return a mutated chromosome (i.e. as in the Matlab introduction).

After completing any Matlab function, you should preferably carry out a *unit test*, i.e. writing a simple wrapper that just provides suitable input to the function in question, and then make sure that the function generates correct output. (You should not hand in any of the unit tests, though).

When you have completed your program, experiment with some different values of the parameters that you are allowed to change, then select a set of parameters (which you may hardcode in `RunSingle.m`), and then carry out 10 runs of the algorithm, noting the values found, both for x_1 and x_2 and for the function $g(x_1, g_2)$ (i.e. the inverse of the fitness value).

b) Since the goal of this problem is for you to familiarize yourselves with GAs, you should also make a parameter search that, in this problem, will concern only the mutation rate (normally, one would of course carry out an analysis involving several (or all) parameters). For this analysis, you should use the `RunBatch.m` file. The skeleton file already carries out a batch of 100 runs with the mutation probability p_{mut} equal to 0.02 ($= 1/m$), and then prints some statistics. You should add code in `RunBatch.m` so that the program makes several batch runs (in sequence) for different values of the mutation probability. Then you should tabulate and plot (and include in your report; see below) the (note!) *median* performance (= fitness value), over the 100 runs, as a function of the mutation probability p_{mut}. Use at least 10 different values of p_{mut} in the range $[0, 1]$, including $p_{\text{mut}} = 0$ and (already implemented) $p_{\text{mut}} = 0.02$. Apart from defining these runs and printing the necessary statistics, do not make *any* other changes in `RunBatch.m`, i.e. keep all other parameters as specified in that file.

c) Note that the GA provides a numerical estimate of the true minimum, but your goal is to find the exact minimum. Thus, guided by the results from part (a), make an educated guess of where the actual minimum $(x_1^*, x_2^*)^{\text{T}}$ might be (hint: If you make long enough runs, the GA will generally get very close to the actual minimum, in most runs), and then prove *analytically* (i.e. by manual calculations, without the help of a computer!) that the point $(x_1^*, x_2^*)^{\text{T}}$ actually is a stationary point of the function g. (You do not need to prove that it is a minimum).

What to hand in You should hand in all Matlab files listed above, including your versions of `RunSingle.m` and `RunBatch.m`. In your report, write a section *Problem1.3* where, for part (a) above, you list the selected parameters (used in `RunSingle.m`) and also include a (neatly formatted, not a Matlab screen dump) table of the values of x_1, x_2, and (note!) $g(x_1, x_2)$ found in your 10 runs. Then, for part (b), include another neatly formatted table showing the values of p_{mut} and the median performance (of the GA) obtained over the corresponding batch runs. Also, include a plot of the median performance as a function of p_{mut}. Then, include a brief discussion of your findings related to the optimal value of p_{mut} for this problem. Finally, for part (c), write down the true minimum $(x_1^*, x_2^*)^{\text{T}}$, as well as the entire derivation, with all relevant intermediate steps, showing that this point is indeed a stationary point. Omitting intermediate steps may result in the solution being returned, with resulting point deductions; see below.

Maximum number of points for this problem: 4p.
Maximum number of points if the problem must be returned for correction: 2p