Stochastic optimization algorithms, problem 1

Adam Tonderski
tadam@student.chalmers.se
930524-1037

Problem 1.1, Penalty Method

In this problem, the task is to use the penalty method to find the minimum of the function:

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

that is subject to the constraint

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

Without any computation it is apparent that $f(x_1, x_2)$ has a minimum $f(1, 2) = 0$. This conclusion is reached by plugging in the values of x_1 and x_2 that make the separate parts reach 0. Since they are both squared the function cannot be negative and $f(1, 2) = 0$ has to be a global minimum.

The first step in solving the constrained problem is defining the function $f_p(\boldsymbol{x}; \mu)$ which is the sum of $f(\boldsymbol{x})$ and the penalty term

$$p(\boldsymbol{x}, \mu) = \mu(\max\{0, (x_1^2 + x_2^2 - 1)\})^2 \tag{3}$$

Combining these two functions gives:

$$f_p(\boldsymbol{x}; \mu) = \begin{cases} (x_1 - 1)^2 + 2(x_2 - 2)^2 + \mu(x_1^2 + x_2^2 - 1)^2 & \text{if } x_1^2 + x_2^2 \geq 1 \\ (x_1 - 1)^2 + 2(x_2 - 2)^2 & \text{otherwise} \end{cases} \tag{4}$$

Gradient descent, which is the algorithm of choice, needs one last ingredient - the gradient. It could be computed numerically, but the function is simple so an analytic solution is straightforward.

$$\nabla f_p(\boldsymbol{x}; \mu) = \begin{cases} \begin{matrix} (2(x_1 - 1) + 4\mu x_1(x_1^2 + x_2^2 - 1))\hat{\mathbf{i}} \\ +(4(x_2 - 2) + 4\mu x_2(x_1^2 + x_2^2 - 1))\hat{\mathbf{j}} \end{matrix} & \text{if } x_1^2 + x_2^2 \geq 1 \\ 2(x_1 - 1)\hat{\mathbf{i}} + 4(x_2 - 2))\hat{\mathbf{j}} & \text{otherwise} \end{cases} \tag{5}$$

The rest of the calculations were performed in Matlab. In order to get a good feeling for the impact of the penalty term the problem was solved for $\mu = 1$, 10, 100, 10^3, 10^4,

10^5. The starting point was chosen to be the minimum of the unconstrained function - $f(1,2) = 0$. Through some experimentation good values for the step size and threshold were found to be 10^-6 and 10^-5 respectively. The step size had to be sufficiently small to avoid overshooting in the case of very large values of μ. These calculations can be reproduced by running the PenaltyMethod script in the attached Matlab files.

Table 1: Results of gradient descent

μ	x_1^*	x_1^*	$\lvert\boldsymbol{x}^*\rvert$
1	0.434	1.210	1.286
10	0.331	0.996	1.049
100	0.314	0.955	1.006
1000	0.312	0.951	1.001
10000	0.312	0.950	1.000
100000	0.312	0.950	1.000

The results in Table 1 show that the results converge towards $(0.312, 0.950)^T$ as μ increases. This point is right on the boundary of the constraint, as can be seen by the absolute value approaching 1, and with lower values for μ the minimum moves outside of the allowed region. This correlates well with the theory, since a very large μ means a very large penalty for staying outside of the allowed region.

Problem 1.2a

We need to find the minimum of

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

In a region that can be defined as

$$x_2, x_1 \in [0, 1] \quad \text{and} \quad x_2 > x_1 \tag{7}$$

First let's check for optima within the region by solving

$$\nabla f(x_1, x_2) = \begin{pmatrix} 8x_1 - x_2 \\ -x_1 + 8x_2 - 6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{8}$$

We get the following system of equations:

$$\begin{cases} 8x_1 - x_2 = 0 \\ -x_1 + 8x_2 - 6 = 0 \end{cases}$$

$$\begin{cases} x_2 = 8x_1 \\ 63x_1 - 6 = 0 \end{cases}$$

$$\begin{cases} x_2 = 8x_1 \\ x_1 = 6/63 \end{cases}$$

$$\begin{cases} x_1 = 2/21 \\ x_2 = 16/21 \end{cases}$$

It's easy to confirm that the found point lies within the allowed region by comparing with (7), and we can compute the value of the optimum

$$f(\frac{2}{21}, \frac{16}{21}) = 4(\frac{2}{21})^2 - \frac{2}{21}\frac{16}{21} + 4(\frac{16}{21})^2 - 6\frac{16}{21} = \frac{16 - 32 + 1024 - 2016}{441} = -\frac{16}{7}$$

If we can prove that $f(x_1, x_2)$ is convex, we have found the global minimum. First we need to calculate the Hessian and it's eigenvalues

$$H = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{pmatrix} = \begin{pmatrix} 8 & -1 \\ -1 & 8 \end{pmatrix} \tag{9}$$

$$|H - \lambda I| = 0 \tag{10}$$

$$(8 - \lambda)^2 - 1 = 0 \tag{11}$$

$$8 - \lambda = \pm 1 \tag{12}$$

$$\lambda = 8 \pm 1 \tag{13}$$

Since both eigenvalues are positive, the function is convex and therefore we have found the global minimum. Most importantly we don't need to check to boundaries!

Problem 1.2b, Lagrange multiplier method

Here we will be using the Lagrange multiplier method to minimize the function

$$f(x_1, x_2) = 15 + 2x_1 + 3x_2 \tag{14}$$

subject to the constraint

$$h(x_1, x_2) = x_1^2 + x_1 x_2 + x_2^2 - 21 = 0 \tag{15}$$

According to the Lagrange multiplier method, we start by forming the function

$$L(x_1, x_2, \lambda) = f(x_1, x_2) + \lambda h(x_1, x_2) = 15 + 2x_1 + 3x_2 + \lambda(x_1^2 + x_1 x_2 + x_2^2 - 21) \tag{16}$$

Then, we set up the three equations that must hold in stationary points

$$\frac{\partial L}{\partial x_1} = 2 + 2\lambda x_1 + \lambda x_2 = 0 \tag{17}$$
$$\frac{\partial L}{\partial x_2} = 3 + \lambda x_1 + 2\lambda x_2 = 0 \tag{18}$$
$$\frac{\partial L}{\partial \lambda} = x_1^2 + x_1 x_2 + x_2^2 - 21 = 0 \tag{19}$$

and solve them

$$(18) - 1.5 * (17) \Rightarrow -2\lambda x_1 + 0.5\lambda x_2 = 0 \tag{20}$$
$$(20)/\lambda \Rightarrow -2x_1 + 0.5x_2 = 0 \tag{21}$$
$$\Leftrightarrow x_2 = 4x_1 \tag{22}$$
$$(22) \text{ and } (19) \Rightarrow x_1^2 + 4x_1^2 + 16x_1^2 - 21 = 0 \tag{23}$$
$$\Leftrightarrow 21x_1^2 = 21 \tag{24}$$
$$\Leftrightarrow x_1 = \pm 1 \text{ and } x_2 = \pm 4 \tag{25}$$

The division by λ in (21) is allowed because $\lambda = 0$ corresponds to an unconstrained problem (it also makes the other equations unsolvable).

In the final step we evaluate the function values in the discovered stationary points

$$f(1, 4) = 15 + 2 + 12 = 29 \tag{26}$$
$$f(-1, -4) = 15 - 2 - 12 = 1 \tag{27}$$

and arrive at the conclusion that $f(-1, -4) = 1$ is the solution!

Problem 1.3, Basic GA program

The task here is to use a genetic algorithm to find the global minimum of

$$g(x_1, x_2) = (1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)) \times \tag{28}$$
$$(30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)) \tag{29}$$

The algorithm was implemented in Matlab and used standard mutations, single point crossover and tournament selection. Initial runs, using parameter values from the example implementation, showed that the system was prone towards premature convergence. Therefore, in the following runs, the parameter values were selected in a way that should explore this behaviour and hopefully give a feeling for how to best avoid it.

Every set of parameters ran 100 times and the results were averaged. The resulting optima, depending on the varying parameters, are shown in Table 2. These calculations can be reproduced by running the FunctionOptimization script in the attached Matlab files. Note however that it is currently set to only run once, so in order to correctly reproduce these results, "numberOfRuns" needs to be changed to 100.

Table 2: Results from running the GA program

N	P_c	P_{mut}	p_{tour}	k_{tour}	$\boldsymbol{x}^*$		$g(\boldsymbol{x}^*)$	
					Avg.	S.D	Avg.	S.D
50	0.8	0.025	0.75	2	$\begin{pmatrix} 0.0878 \\ -0.9414 \end{pmatrix}$	$\begin{pmatrix} 0.3852 \\ 0.2571 \end{pmatrix}$	7.0953	17.9456
50	0.8	0.1	0.75	2	$\begin{pmatrix} 0.0002 \\ -1.0001 \end{pmatrix}$	$\begin{pmatrix} 0.0058 \\ 0.0053 \end{pmatrix}$	3.0190	0.0354
100	0.8	0.1	0.75	2	$\begin{pmatrix} 0.0003 \\ -1.0008 \end{pmatrix}$	$\begin{pmatrix} 0.0058 \\ 0.0069 \end{pmatrix}$	3.0254	0.0428
20	0.8	0.1	0.75	2	$\begin{pmatrix} 0.0025 \\ -0.9992 \end{pmatrix}$	$\begin{pmatrix} 0.0136 \\ 0.0148 \end{pmatrix}$	3.1213	0.4454
50	0.99	0.01	0.75	2	$\begin{pmatrix} 0.0745 \\ -0.8983 \end{pmatrix}$	$\begin{pmatrix} 0.4416 \\ 0.2985 \end{pmatrix}$	9.4420	20.3370
50	0.99	0.01	0.75	3	$\begin{pmatrix} 0.1741 \\ -0.7703 \end{pmatrix}$	$\begin{pmatrix} 0.6665 \\ 0.4235 \end{pmatrix}$	16.3982	27.5377
20	0.99	0.01	0.75	3	$\begin{pmatrix} 0.4798 \\ -0.5006 \end{pmatrix}$	$\begin{pmatrix} 1.0160 \\ 0.6001 \end{pmatrix}$	76.0161	164.3764

All simulation used a chromosome length of 50 (25 genes per variable) and only one individual was carried over without modification each generation. As for the number of generations, it was set to 200. This is a relatively high number (especially compared to the example implementation), and this was useful since the primary area of interest was premature convergence. With this high a number the parameter combinations that 'got stuck' had ample time to escape their local minima but were not able to do so.

On the other hand, this high value could make some parameter combinations that are very slow and inefficient look really good. This is especially true of the runs with a high mutation probability, since if the number of generations goes to infinity, random mutations will find the optimum sooner or later. Ideally, the number of generations would have been varied as well to detect this kind of bias but that is outside of the scope of this assignment.

The conclusions that can be drawn from the results are that it is dangerous to keep the mutation probability too low, especially in combination with a high crossover ratio. A large tournament size can also contribute to premature convergence, since it increases the bias towards better individuals. This might seem very obvious, since this is exactly what the theory predicts, but it is always good to confirm it with some actual simulations. Another interesting observation that can be made is that advantage of having a large population can be severely offset by tuning the other parameters. The runs with $p_{mut} = 0.1$, $p_c = 0.8$, $p_{tour} = 0.75$ and $k_{tour} = 2$ came very close to the optimum with just $N = 20$, and any additional population after 50 seems to do no good (and maybe even a little bit of harm, although the difference is way too small to tell). However, in the runs with the 'worse' set of parameters, going from 20 to 50 individuals improved the results tremendously.

Finally, it's important to note that the actual values that proved to be good for optimizing this function cannot necessarily be generalized to other problem. The methodology for choosing parameters, should work decently though. If we converge prematurely we can try increasing the mutation rate, adding more individuals to the population and even lowering the crossover rate. And the other way around, if the algorithm is finding the optimum, but doing so very slowly, it could be worth trying the opposite operations.

Analytic proof

To prove analytically that the discovered point is in fact an optimum, we can calculate the gradient of g, insert the point and verify that the value is $(0, 0)$. Since the function is quite long and complicated we will divide it into several smaller functions and calculate those derivatives separately.

$$\nabla g(\boldsymbol{x}) = \nabla(g_1(\boldsymbol{x})g_2(\boldsymbol{x})) = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} g_2 + \frac{\partial g_2}{\partial x_1} g_1 \\ \frac{\partial g_1}{\partial x_2} g_2 + \frac{\partial g_2}{\partial x_2} g_1 \end{pmatrix} \tag{30}$$

$$\tag{31}$$

where

$$\begin{aligned}
g_1 =& 1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2) \\
\frac{\partial g_1}{\partial x_1} =& 2(x_1 + x_2 + 1)(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2) + \\
& + (x_1 + x_2 + 1)^2(-14 + 6x_1 + 6x_2) \\
\frac{\partial g_1}{\partial x_2} =& 2(x_1 + x_2 + 1)(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2) + \\
& + (x_1 + x_2 + 1)^2(-14 + 6x_1 + 6x_2) \\
g_2 =& 30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2) \\
\frac{\partial g_2}{\partial x_1} =& 4(2x_1 - 3x_2)(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2) + \\
& + (2x_1 - 3x_2)^2(-32 + 24x_1 - 36x_2) \\
\frac{\partial g_2}{\partial x_2} =& 6(2x_1 - 3x_2)(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2) + \\
& + (2x_1 - 3x_2)^2(48 - 36x_1 + 54x_2)
\end{aligned}$$

Now we can insert the value for $x_1 = 0$ and $x_2 = -1$ that we found in our simulations.

Using $(x1 + x2 + 1) = 0$ makes the calculations very easy.

$$\begin{aligned}
g_1(0,-1) =& 1 \\
\frac{\partial}{\partial x_1} g_1(0,-1) =& 0 \\
\frac{\partial}{\partial x_2} g_1(0,-1) =& 0 \\
g_2(0,-1) =& 30 + 3^2 \cdot (18 - 48 + 27) = 30 + 9 \cdot (-3) = 33 \\
\frac{\partial}{\partial x_1} g_2(0,-1) =& 4 \cdot 3 \cdot (-3) + 9 \cdot (-32 + 36) = -36 + 36 = 0 \\
\frac{\partial}{\partial x_2} g_2(0,-1) =& 6 \cdot (-3) \cdot (-3) + 9 \cdot (-6) = 27 - 27 = 0
\end{aligned}$$

and finally we get

$$\nabla g(0,-1) = \begin{pmatrix} 0 \cdot 33 + 0 \cdot 1 \\ 0 \cdot 33 + 0 \cdot 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{32}$$

which is what we wanted to prove!