Problem Set 5: Optimization

Jacob Adenbaum and Albert Rodriguez-Sala
Programming for Economics – ECNM10106
The University of Edinburgh

Spring 2024

Due Monday, March 4, 2024

Instructions: Attempt all the exercises. Partial credit will be awarded for incomplete solutions which demonstrate some understanding. You should always try to explain your results, and provide as much intuition as possible. Pay careful attention to how you present your results. Be sure to take some time to make them easy to read and to understand. Avoid printing out irrelevant output, and pay attention to how you construct your figures and tables. You will be assessed both on the correctness of your results, and on how easy you make it to understand them.

Constrained Rosenbrock Problem

Let's revisit the classic Rosenbrock function from last week. Recall that this $f : \mathbb{R}^n \to \mathbb{R}$ is defined as

$$f(x) = \sum_{i=1}^{n-1} \left(a(1 - x_i)^2 + b\big(x_{i+1} - x_i^2\big)^2 \right)$$

for parameters a and b. It is particularly well known as a test function for optimization, with many local minima. On Problem Set 4, we explored various optimizers and how well they were able to find the *unconstrained* minimum of this function. Now, consider the constrained minimization problem

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \sum_{i=1}^{n-1} \left(a(1 - x_i)^2 + b\big(x_{i+1} - x_i^2\big)^2 \right) \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i^2 = r \end{aligned} \tag{1}$$

where we constrain x to lie on the phere of radius $\sqrt{r}$. Note that when $r = n$, the global minimum of this function lies in the constraint set.

(a) **Sequential Quadratic Programming with Newton's Method:** Recall that SQP is a method for constrained optimization that attempts to solve the system of first order conditions of the Lagrangian directly as a root finding problem.

(i) Write down the Lagrangian[1] for the problem and write a function that calculates the gradient $D\mathcal{L}(x, \lambda)$.[2] Now, test out your gradient on several different points (to be comprehensive, try sampling 100 points uniformly on the unit square $[0, 1]^n$ and calculate the root mean squared

[1] When you write the function for this as well, you will probably want to structure it to take a single argument: a vector containing both x and λ stacked together

[2] You can either use a package for autodifferentiation, or you can calculate the gradient by hand.

error[3]), and compare the results to what you would get if you use finite differences. Set $a = 1$, and compare the results when $b = 1$ as well vs. when $b = 100$. What about if $b = 1000$? Is there a value of b where finite differences stop working well? Explain.

(ii) Implement Newton's method yourself, and then use it to solve the constrained Rosenbrock problem with sequential quadratic programming.[4] Use a stopping criterion that stops the algorithm when the norm of the gradient of the objective (in this case, the Lagrangian) falls below a tolerance level ϵ. Try it out for a simple example. Set $n = 2$, $a = b = 1$, and $r = 2$. Start with $x_0 = (0, 0)$ and $\lambda_0 = 0$. Make a plot showing the progress of the algorithm (plot the values of x^k at each step, the constraint set, and the contours of f). Use a tolerance of $\epsilon = 10^{-6}$. How is the algorithm doing?[5]

(iii) Keep $r = 2$, and move around your initial starting point x_0. How sensitive are your results to the initial guess? We know that in this case, the true minimum is at x such that $x_i = 1$ for all i. How close do you need to make x_0 to the true minimum to ensure convergence?

(iv) Try out your function for a variety of parameters and report the results. How is the algorithm doing? Is it reliably finding the correct answer? How many function evaluations does it require? Try your best to do this systematically.

(b) **Penalty Method:** Recall that the penalty method works by solving a sequence of augmented problems

$$\min_{x \in \mathbb{R}^n} f(x) + P_k ||g(x)||^2 \tag{2}$$

where P_k controls the strength of the penalty for violating the constraint.

(i) Try solving the problem directly when $a = b = r = 10$, $n = 10$, and $P_k = 10^{12}$. Start with $x_0 = 0$. Use your favorite algorithm for the unconstrained problem. (Here it's okay to use someone else's implementation). How well does this fare? Do you converge? If you do, how long does it take (how many function evaluations)?

(ii) Now try starting with $P_0 = 1$, and solve a sequence of problems. Use the solution at k as a "warm-start" for the $k + 1$th problem. Set $P_{k+1} = 10 \times P_k$ at each iteration. Repeat until you reach $P_{12} = 10^{12}$. How do your results compare? Do you converge to the true minimum? If not, try again with a better initial guess (remember, in this case you know where the true minimum is). Does it improve?

(iii) Repeat with $b = 1000$. Does anything change?

(iv) What about with $r = 1$?

[3]Set $n = 5$, and draw each (x^k, λ^k) as 6 random uniform numbers (for $k = 1, \ldots 100$). For each point (x^k, λ^k), let $y^k = D\mathcal{L}(x^k, \lambda^k)$, and let $\hat{y}^k$ be the finite differences approximation. The root mean squared error is defined as

$$RMSE = \sqrt{\frac{1}{100} \sum_{k=1}^{100} ||y^k - \hat{y}^k||^2}$$

Remember, of course, that the norm of a vector z is defined as $||z||^2 = z^T z$

[4]A reminder: never invert the Hessian directly. Always use a linear solver. I.e, for the system of linear equations $Ax = b$ the syntax is `A\b`

[5]Hint: If you get an error about a singular exception, this means that the Hessian is not invertible. Think about what that means, and propose a solution. You may need to consider a different initial guess

(c) **Augmented Lagrangian:** Now, let's try implementing the Augmented Lagrangian Method. Recall that this approach involves solving a sequence of problems

$$\begin{aligned} x_k &= \arg\min_{x\in\mathbb{R}^n} \quad f(x)+\frac{P_k}{2}||g(x)||^2 + \lambda_k \cdot g(x) \\ P_{k+1} &= \alpha P_k \\ \lambda_{k+1} &= \lambda_k + P_k g(x_k) \end{aligned}$$

(i) Start with $P_0 = 1$ and $\alpha = 10$. Use $a = b = r = 10$ and $n = 10$. Guess $x_0 = 0$, use your favorite algorithm for the inner unconstrained problem, and iterate until $||\lambda_{k+1} - \lambda_k||^2 < 10^{-8}$. Report your results.

Do you find the true minimum? How large does P_k need to be when λ converges?

(ii) Compare the performance of the Augmented Lagrangian to the standard Penalty Method. Try it for a variety of parameter values, and present your results in a nice way. What do you conclude from this?

(d) **Parameter Transformation:** This particular constraint set (the unit sphere) is fairly easy to parameterize directly. Consider the mapping $h : \mathbb{R}^{n-1} \to S^n$ into the unit sphere [6]:

$$h_i(z) = \begin{cases} \left(\frac{\exp(z_i)}{1+\sum_{s=1}^{n-1}\exp(z_s)}\right)^{\frac{1}{2}} & \text{if } i \leq n-1 \\ \left(\frac{1}{1+\sum_{s=1}^{n-1}\exp(z_s)}\right)^{\frac{1}{2}} & \text{if } i = n \end{cases} \tag{3}$$

whose inverse is given by

$$h_i^{-1}(x) = 2\log\left(\frac{x_i}{x_n}\right) \text{ for } i = 1, \ldots, n-1 \tag{4}$$

Try solving the constrained optimization problem directly by using this re-parameterization. Compare to parts (a) - (c). How do your results compare? How many function evaluations does it require? Does it require more or less time to compute? What do you take away from this?

[6] Note: S^n is notation for the unit sphere in $\mathbb{R}^n$ (the set of vectors whose length is 1). It is defined as

$$S^n = \left\{ x \in \mathbb{R}^n : \sum_{i=1}^{n} x_i^2 = 1 \right\}$$

Remember, if you know how to map into the unit sphere, you can always get a sphere with a different radius (just use $rh(z)$ instead of $h(z)$)