MAT 3007 Optimization Homework 7
No Submission is required
Solution

1. Consider the following optimization

$$\begin{array}{ll} \text{minimize} & x_1^2 + x_2^2 \\ \text{subject to} & (x_1-1)^2 + (x_2-1)^2 \le 1 \\ & (x_1-1)^2 + (x_2+1)^2 \le 1 \end{array}$$

with variable $x \in \mathbb{R}^2$.

(a) Find the optimal point $x^\star$ and optimal value $p^\star$.
There is only one feasible solution, $(1,0)$, so it is optimal for the primal problem, and we have $p^* = 1$.

(b) Give the KKT conditions. Do there exist Lagrange multipliers that prove that $x^\star$ is optimal?
The KKT conditions are

$$(x_1-1)^2 + (x_2-1)^2 \le 1, \quad (x_1-1)^2 + (x_2+1)^2 \le 1,$$

$$\lambda_1 \ge 0, \quad \lambda_2 \ge 0,$$

$$2x_1 + 2\lambda_1(x_1-1) + 2\lambda_2(x_1-1) = 0,$$

$$2x_2 + 2\lambda_1(x_2-1) + 2\lambda_2(x_2+1) = 0,$$

$$\lambda_1((x_1-1)^2 + (x_2-1)^2 - 1) = \lambda_2((x_1-1)^2 + (x_2+1)^2 - 1) = 0.$$

At $x = (1,0)$, these conditions reduce to

$$\lambda_1 \ge 0, \quad \lambda_2 \ge 0, \quad 2 = 0, \quad -2\lambda_1 + 2\lambda_2 = 0,$$

which (clearly, in view of the third equation) have no solution.

(c) Derive and solve the Lagrange dual problem. Does strong duality hold?
The Lagrange dual function is given by

$$g(\lambda_1,\lambda_2) = \inf_{x_1,x_2} L(x_1,x_2,\lambda_1,\lambda_2)$$

where

$$\begin{aligned} L(x_1,x_2,\lambda_1,\lambda_2) &= x_1^2 + x_2^2 + \lambda_1((x_1-1)^2 + (x_2-1)^2 - 1) + \lambda_2((x_1-1)^2 + (x_2+1)^2 - 1) \\ &= (1+\lambda_1+\lambda_2)x_1^2 + (1+\lambda_1+\lambda_2)x_2^2 - 2(\lambda_1+\lambda_2)x_1 - 2(\lambda_1-\lambda_2)x_2 + \lambda_1 + \lambda_2. \end{aligned}$$

L reaches its minimum for

$$x_1 = \frac{\lambda_1+\lambda_2}{1+\lambda_1+\lambda_2}, \quad x_2 = \frac{\lambda_1-\lambda_2}{1+\lambda_1+\lambda_2},$$

and we find

$$g(\lambda_1,\lambda_2) = \begin{cases} -\frac{(\lambda_1+\lambda_2)^2+(\lambda_1-\lambda_2)^2}{1+\lambda_1+\lambda_2} + \lambda_1+\lambda_2 & \text{if } 1+\lambda_1+\lambda_2 \geq 0, \\ -\infty & \text{otherwise,} \end{cases}$$

where we interpret $a/0 = 0$ if $a = 0$ and as $-\infty$ if $a < 0$. The Lagrange dual problem is given by

$$\begin{array}{ll} \text{maximize} & \dfrac{\lambda_1+\lambda_2-(\lambda_1-\lambda_2)^2}{1+\lambda_1+\lambda_2} \\ \text{subject to} & \lambda_1,\lambda_2 \geq 0. \end{array}$$

Since g is symmetric, the optimum (if it exists) occurs with $\lambda_1 = \lambda_2$. The dual function then simplifies to

$$g(\lambda_1,\lambda_1) = \frac{2\lambda_1}{2\lambda_1+1}.$$

We see that $g(\lambda_1,\lambda_2)$ tends to 1 as $\lambda_1 \to \infty$. We have $d^* = p^* = 1$, but the dual optimum is not attained.

Recall that the KKT conditions only hold if (1) strong duality holds, (2) the primal optimum is attained, and (3) the dual optimum is attained. In this example, the KKT conditions fail because the dual optimum is not attained.

2. Let $a \in \mathbb{R}^n \setminus \{0\}$ and $b \in \mathbb{R}$ be given and define the hyperplane $C := \{x \in \mathbb{R}^n : a^\top x = b\}$. Compute the projection $x \in \mathbb{R}^n$ onto the set C, i.e., find the optimal solution to the following optimization problem:

$$\min_{y\in\mathbb{R}^n} \frac{1}{2}\|y-x\|^2 \quad \text{subject to} \quad a^\top y = b.$$

3. Consider the optimization problem:

$$\min_x \; 4x^2 - x^4.$$

Starting from the interval $[-1, 1/2]$, solve the problem using both the Bisection Method and the Golden Section Method in MATLAB or Python. For both methods, output the midpoint of the final interval as the approximate solution. Compare the number of iterations required by each method to achieve a solution that is within 10^{-4} of the optimal solution in the given starting interval, $x^* = 0$.

Refer to the code for details.
After 15 iterations of the bisection method, it returns a solution $x = -1.52587891 * 10^{-5}$ and the distance between left and right bounds is $9.16 * 10^{-5}$.

After 20 iterations of the golden section method, it returns a solution $x = 1.65267403 * 10^{-5}$ and the distance between left and right bounds is $9.92 * 10^{-5}$.

4. If you have observations $\{x_1, ..., x_n\}$ that follow Poisson distribution, to estimate the parameters in Poisson, you can derive out the likelihood function:

$$p(x_1, ..., x_n) = \prod_{i=1}^{n} e^{-\mu} \frac{\mu^{x_i}}{x_i!}.$$

Now you want to use the maximum (log-)likelihood estimation (MLE) to find out the best parameter μ, the MLE optimization model is

$$\text{(MLE)} \quad \max_{\mu} \ f(\mu) = \log \ p(x_1, ..., x_n)$$

where log function has base of e (i.e. same as ln function).

(a) Write down the main step in each iteration of the gradient descent algorithm for solving the above MLE optimization model.
After simplication, we have

$$f(\mu) = -n\mu + \sum_{i=1}^{n} (x_i \log \mu - \log(x_i!)),$$

$$\frac{df(\mu)}{d\mu} = -n + \sum_{i=1}^{n} \frac{x_i}{\mu}.$$

So the main step in gradient descent is

$$\mu^{k+1} = \mu^k + \alpha^k \left(-n + \sum_{i=1}^{n} \frac{x_i}{\mu^k} \right).$$

(b) Write down the main step in each iteration of the Newton's method for solving the above MLE optimization model.
We have

$$\frac{d^2 f(\mu)}{d^2 \mu} = -\sum_{i=1}^{n} \frac{x_i}{\mu^2}$$

So the main step in Newton's method is

$$\mu^{k+1} = \mu^k - \alpha^k \frac{-n + \sum_{i=1}^n \frac{x_i}{\mu^k}}{-\sum_{i=1}^n \frac{x_i}{(\mu^k)^2}} = \mu^k - \alpha^k \mu^k \frac{n\mu^k - \sum_{i=1}^n x_i}{\sum_{i=1}^n x_i}$$

(c) Will gradient descent converge to a local optimum or a global optimum in the above problem? Explain your answer.
We have

$$\frac{d^2 f(\mu)}{d\mu^2} = -\sum_{i=1}^n \frac{x_i}{\mu^2} \le 0,$$

since the Poisson-distributed values satisfy $x_i \ge 0$ and $\mu > 0$. Therefore, the function $f(\mu)$ is concave. Maximizing a concave function is equivalent to minimizing a convex function, which implies that this is a convex optimization problem. Thus the graident descent converges to global optimum.

5. Implement the gradient descent method in Matlab to solve the optimization problem:

$$\min_{x \in \mathbb{R}^2} f(x) = e^{1-x_1-x_2} + e^{x_1+x_2-1} + x_1^2 + x_1 x_2 + x_2^2 + x_1 - 3x_2.$$

The following input parameters should be considered:

- $x^0 = (0,0)^\top$ – the initial point.
- $\epsilon = 10^{-5}$ – the tolerance parameter. The method should stop whenever the current iterate x^k satisfies the criterion $\|\nabla f(x^k)\| \le \epsilon$.
- $\sigma, \gamma = \frac{1}{2}$ – parameters for backtracking and the Armijo condition.

Implement different methods using the given parameter.

(a) Implement the gradient descent method using constant stepsize. Try two different constant stepsizes $\alpha_1 = 1$ and $\alpha_2 = 0.1$. Report the number of iterations, the final objective function value, and the point to which the methods converged, respectively.
Refer to the code for details. With $\alpha_1 = 1$, the algorithm diverges. With an approriate stepsize $\alpha_2 = 0.1$, the algorithm converges after 120 iterations, with $x^* = [-1.571284, 2.428703]$, $||\nabla f(x^*)|| = 9.133687\text{e-}06$, $f(x^*) = -2.285680$. The path is show in Figure 1.

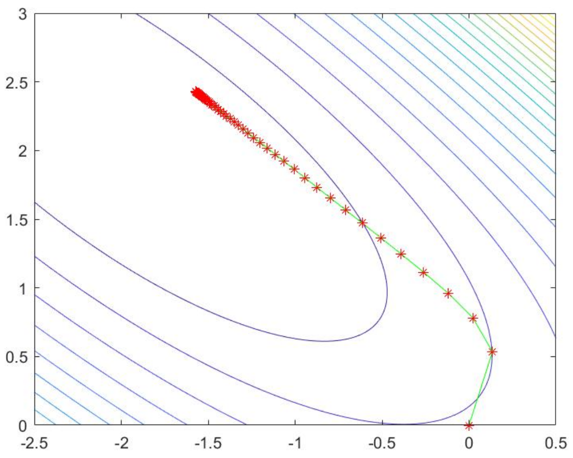

Figure 1: Solution Path for Constant Stepsize

(b) Implement the gradient descent method using backtracking / Armijo line search method, report the number of iterations, the final objective function value, and the point to which the methods converged.
Armijo backtracking converges after 41 iterations, with $x^* = [-1.571286, 2.428704]$, $||\nabla f(x^*)|| =$ 8.269390e-06, $f(x^*) = -2.285680$. The paths is shown in Figure 2.

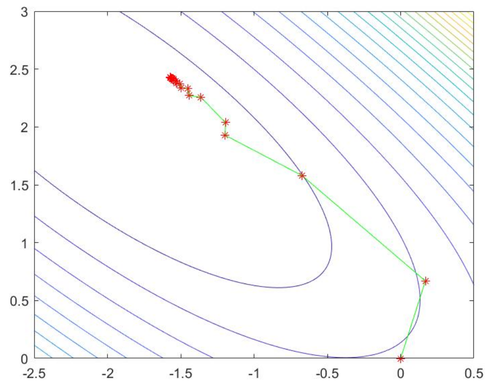

Figure 2: Solution Path for Armijo Backtracking Stepsize

(c) Plot figures of the solution path for each of the different step size strategies (similar to the one in the lecture slides).
See Figure 1 and 2.

Note: You can also use Python to implement if you want. We have provided a prototype code in Matlab format in the folder `Code_prototype`. You need to fill the parts marked as "**TODO**" by yourself. You can also write the code from scratch if you want.

6. Implement the Newton's method in Matlab to solve the same problem in Problem 3. Use also the same parameter setup in Problem 5. However, use only Armijo backtracking line search for choosing stepsize in this question. Report the number of iterations, the final objective function value, the point to which the methods converged. Plot figures of the solution path.

   Note: You can also use Python to implement if you want. We have provided a prototype code in Matlab format in the folder `Code_prototype`. You need to fill the parts marked as "**TODO**" by yourself. You can also write the code from scratch if you want.

   Refer to the code for details

   Newton's method converges after 3 iterations, with

$$x^* = [-1.571290, 2.428710],\ \|\nabla f(x^*)\| = 2.167840\text{e-}07,\ f(x^*) = -2.285680.$$

   The path of Newton's method is shown in Figure 3.

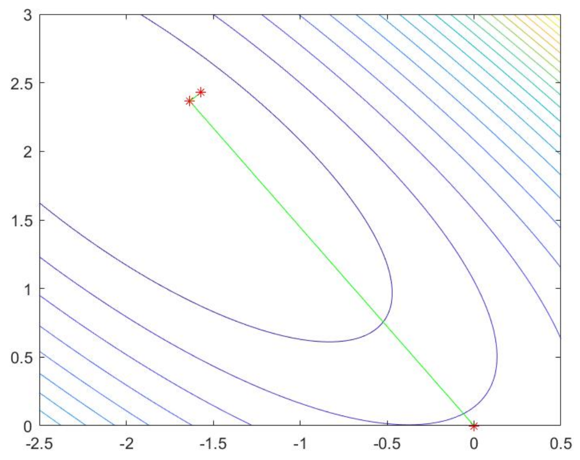

Figure 3: Solution Path for Newton's Method