Homework 3

Stats 102B Lec 1 and 2

Spring 2024

General Guidelines

Show all your work, including any and all relevant code and output. Any and all collaboration must adhere to **Level 1** collaboration described in the Stats 102B Collaboration Policy.

Please submit your homework as a single file, in PDF format only. Name your assignment file with the convention `123456789_stats102b_hw0.pdf`, where `123456789` is replaced with your UID and `hw0` is updated to the actual homework number. Include your first and last name and UID in your assignment as well.

All R code is expected to follow the Tidyverse Style Guide: https://style.tidyverse.org/. If you scan or take a picture of any written work, please make sure the resolution is high enough that your work is clear and legible. Submissions with severe style or formatting issues may receive a penalty. Any submission that cannot be properly read will not be graded.

Question 1

(a)

Write a function `newts_method(g, grad_g, hess_g, w0, K)` that runs Newton's Method with the following parameters:

- `g` is a function $g : \mathbb{R} \to \mathbb{R}$
- `grad_g` is the gradient of g, a function $\nabla g : \mathbb{R}^n \to \mathbb{R}^n$
- `hess_g` is the Hessian of g, a function $\nabla g : \mathbb{R}^n \to \mathbb{R}^{n \times n}$
- `w0` is the initial point
- `K` is the number of iterations

The output should be a list object with the following components:

- `$index` should represent w^*, the local minimum
- `$val` should represent $g(w^*)$, the value of the function at the local minimum

(b)

Implement Newton's Method for $g(w) = e^{-w_1^2 w_2} + \sin(w_2)$ with $w_0 = (1, -5)^T$ and $K = 10$ (wow, look at that small number of iterations). Explain why we can generally reach convergence faster than gradient descent, but explain the trade-off.

(c)

Recall the function below from Homework 2:

$$g(w_1, w_2) = w_1^8 + w_2^8$$

Run Newton's Method with $w_0 = (1, -1)^T$ and $K = 1000$. You should get an error. Explain what it means.

(d)

Modify your Newton's Method function above to run the Regularized Newton's Method with $\varepsilon = 10^{-7}$, $w_0 = (1, -1)^T$, and $K = 1000$, and show the `$index` of your output list.

(e)

Vary the ε parameter and report your findings. You may consider $\varepsilon = 1$ all the way to $\varepsilon = 10^{-128}$.

Question 2

Much like the numerical differentiation formulas provided in lecture, there are also numerical differentiation formulae for second-order partial derivatives:

$$f_{xx}(x, y) \approx \frac{f(x+h, y) - 2f(x, y) + f(x-h, y)}{h^2}$$

$$f_{yy}(x, y) \approx \frac{f(x, y+k) - 2f(x, y) + f(x, y-k)}{k^2}$$

$$f_{xy}(x, y) \approx \frac{f(x+h, y+k) - f(x+h, y-k) - f(x-h, y+k) + f(x-h, y-k)}{4hk}$$

(a)

Use these centered difference formulas above to modify your Newton's Method function to numerically calculate both the gradient and Hessian at each step. More specifically write a function `newts_method_num(g, h, k, w0, K)` that runs Newton's Method with a numerical gradient and Hessian with the following parameters:

- `g` is a function $g : \mathbb{R} \to \mathbb{R}$
- `h` is a small step size in the x direction
- `k` is a small step size in the y direction
- `w0` is the initial point
- `K` is the number of iterations

The output should be a list object with the following components:

- `$index` should represent w^*, the local minimum
- `$val` should represent $g(w^*)$, the value of the function at the local minimum

Before moving on, it may be helpful to check your work. Homework 2 has several good examples.

(b)

Recall that the probability density function of a beta distribution with parameters $\alpha, \beta > 0$ is defined by

$$f(x|\alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\mathrm{B}(\alpha, \beta)}$$

where

$$\mathrm{B}(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$$

is the beta function.

To compute the maximum likelihood estimator, you may be tempted (again, sorry we borrowed this from the last homework) to compute the log-likelihood and take partial derivatives with respect to α and β and attempt to solve a system of equations as you usually do. However, it turns out that there is no closed-form solution. You may be tempted to try Newton's Method, but try calculating the gradient or worse the Hessian of a function that has the gamma function. Instead, we will try to compute a maximum likelihood estimate by using your numerical Newton's Method's written above.

Let $x_1, x_2, \ldots, x_n$ be independent and identically distributed values from a Beta(α, β) distribution. Show that the negative log-likelihood of α and β can be written as

$$-\log L(\alpha, \beta) = -\left[(\alpha - 1)\sum_{i=1}^{n} \log(x_i) + (\beta - 1)\sum_{i=1}^{n} \log(1 - x_i) - N \log \mathrm{B}(\alpha, \beta)\right]$$

(c)

Read in the `.rds` object from Bruin Learn which contains 1000 observed values from an unknown beta distribution. First write a function for the negative log-likelihood above, then run your algorithm on the negative log-likelihood with $w_0 = (1, 1)^T$, $K = 100$, $h = 0.0001$, $k = 0.0001$, and show the `$index` of your output list.

Hint: You can use the `readRDS()` function to read in `.rds` files, and the `beta()` function to compute $\mathrm{B}(\alpha, \beta)$.

Question 3

Show that Symmetric Rank One (SR1) method for the inverse Hessian takes the form

$$H^k = H^{k-1} + \frac{(x^k - H^{k-1}y^k)(x^k - H^{k-1}y^k)^T}{(x^k - H^{k-1}y^k)^T y^k}$$

where $x^k := w^k - w^{k-1}$ and $y^k := \nabla g(w^k) - \nabla g(w^{k-1})$.

Question 4

(a)

Write a function `bfgs(g, grad_g, w0, K)` that runs the BFGS algorithm with the following parameters:

- `g` is a function $g : \mathbb{R} \to \mathbb{R}$
- `grad_g` is the gradient of g, a function $\nabla g : \mathbb{R}^n \to \mathbb{R}^n$
- `w0` is the initial point
- `K` is the number of iterations

The output should be a list object with the following components:

- `$index` should represent w^*, the local minimum
- `$val` should represent $g(w^*)$, the value of the function at the local minimum

(b)

Consider the function: $f(x, y) = (a - x)^2 + b(y - x^2)^2$. If $a = 1, b = 100$, find the global minimum via BFGS. Try different values of K and different starting points w_0.

(c)

Next, assume we have an n-dimensional function with

$$f(\boldsymbol{x}) = \frac{1}{2}\sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n-1} \frac{1}{i+1}(x_i + x_{i+1})$$

where $\boldsymbol{x} = (x_1, \ldots, x_n)$. For $n = 3$, calculate the gradient of the function and then find the global minimum using your `bfgs()` function.

(d)

Compare your results above with the `optim()` function with `method = "BFGS"`.