Introduction to Optimization
HW 1

Numerical and analytical differentiation of function, Gradient and Hessian.

Before reading this document please watch the video Homework on analytical and numerical computation of gradient and Hessian

This exercise consists of three parts:

In the first part we will obtain a closed formula for the Gradient and the Hessian of a function.

In the second part we will use Matlab to compute the numeric differentiation and verify the correctness of the analytical formulas developed in the first part.

In the third part we will compute the gradient of a feed forward neural network.

1 ANALYTICAL DIFFERENTIATION

In the first part we will obtain a closed formula for the Gradient and the Hessian of some general functions.

Task 1

For $f1(x) = \varphi(Ax)$ where:

$f1 : \mathbb{R}^n \to \mathbb{R}$ Nonlinear multivariate function.

$x \in \mathbb{R}^{n\times 1}$ Function argument, vector.

$A \in \mathbb{R}^{m\times n}$ Some arbitrary matrix.

$\varphi : \mathbb{R}^m \to \mathbb{R}$ Nonlinear multivariate function.

Develop a formula for the Gradient and Hessian of $f1(x)$ given the gradient and hessian of φ.

Task 2

For $f2(x) = h(\varphi(x))$ where:

$f2 : \mathbb{R}^n \to \mathbb{R}$ Nonlinear multivariate function.

$x \in \mathbb{R}^{n\times 1}$ Function argument, vector.

$\varphi : \mathbb{R}^n \to \mathbb{R}$ Nonlinear multivariate function.

$h : \mathbb{R} \to \mathbb{R}$ Nonlinear scalar function.

Develop a formula for the gradient and hessian of the function $f2$ given:

Introduction to Optimization
HW 1

Gradient and Hessian of φ and the first and second derivatives of h.

Task 3

In this task we will write a Matlab function that implements task 1 and task 2.

1. For the following:

$$\varphi\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}\right) = \sin(x_1 x_2 x_3)$$

Write a function that takes x as an input and returns the value, the gradient and the hessian of φ in point x.

$$h(x) = \exp(x)$$

Write a function that takes x as an input and returns the value, the first and second derivatives of h in point x

2. Create a Matlab function "f1" which receives :
   - Input vector $x \in \mathbb{R}^3$
   - Matlab structure array (use Matlab struct command)
     The Matlab structure will contain whatever parameters and functions you may need such as Matrix A, and the value, gradient and hessian function of φ (check how to pass this functions inside struct) etc.

   The function "f1" **analytically** computes the function value, Gradient and Hessian at point x according to the formulas you've developed in tasks 1.

   Note: Generate A to be 3 by three magic matrix, this matrix can be generated using the following Matlab command: `A = magic (3)`

3. Create a Matlab function "f2" which receives :
   - Input vector $x \in \mathbb{R}^3$
   - Matlab structure array (use Matlab struct command)
     The Matlab structure will contain derivatives of h and the gradient and hessian function of φ and whatever other parameters and functions you may need in order to do the task.

   The function "f2" **analytically** computes the function value, Gradient and Hessian at point x according to the formulas you've developed in tasks 2.

Introduction to Optimization
HW 1

For clarification “f1” and “f2” should be implemented as follows

<u>Function signature:</u>

$$[f,\ g,\ H] = f(x,\ par)$$

<u>Function inputs:</u>
x - Input vector, the point on which we calculate the function outputs.
par - All additional parameters for the computation (for example: matrix A or ε for the following task).

<u>Function outputs:</u>
f - Function value on point x, $f1(x)$ for task 1 and $f2(x)$ for task 2.
g - Gradient of function on point x, $\nabla f1(x)$ for task 1 and $\nabla f2(x)$ for task 2.
H - Hessian of function on point x, $\nabla^2 f1(x)$ for task 1 and $\nabla^2 f2(x)$ for task 2.

Important note: don’t compute unnecessary things! Use the Matlab function "nargout" to check the number of requested variables before you compute anything. For example if you call the function like this: $fval = f(x,\ par)$, you only need to compute function value and there is no need to waste time over computing g or H.

Introduction to Optimization
HW 1

2 NUMERICAL DIFFERENTIATION

In this part we will compute the Gradient and Hessian of a function **numerically** using Matlab and use the numerical computation to verify our results from tasks 1 and 2.

Gradient computation

We can compute the Gradient of a function f on point x numerically, using the function value at nearby points by computing the differences:

$$g(x) = \begin{bmatrix} \dfrac{f(x+\varepsilon e_1) - f(x-\varepsilon e_1)}{2\varepsilon} \\ \dfrac{f(x+\varepsilon e_2) - f(x-\varepsilon e_2)}{2\varepsilon} \\ \dots \\ \dfrac{f(x+\varepsilon e_n) - f(x-\varepsilon e_n)}{2\varepsilon} \end{bmatrix}$$

$x \in \mathbb{R}^n$

$e_i = [a_1, \quad a_2, \quad \dots \quad a_n]; a_{j \neq i}, a_i = 0$

$\varepsilon_{machine} = 2 \cdot 10^{-16}$

$\varepsilon = (\varepsilon_{machine})^{(1/3)} \cdot max(abs(x))$

Hessian computation

We compute the Hessian in a much faster way than the analytical one (explain to yourself why) by using this method:

$H^{(i)}(x) \approx \frac{1}{2\varepsilon}\left(g\left(x+\varepsilon e_i\right) - g\left(x-\varepsilon e_i\right)\right)$ Where $H^{(i)}$ is the ith column of H .

Note: you should use the analytical g to compute H faster.

Introduction to Optimization
HW 1

Task 4

Create a Matlab function "numdiff" which **numerically** computes the Gradient and Hessian.

<u>Function signature:</u>

$$[gnum, Hnum] = numdiff\ (@myfunc\ ,x\ ,par\ )$$

<u>Function inputs:</u>
$@myfunc$ - function handle, will hold the functions from task 3 ("f1" or "f2"), to obtain the function value and analytical gradient if needed.
x - Input vector, the point on which we calculate the function outputs.
par - All additional parameters for the computation for example ε (increment of x) for the following task.

<u>Function outputs:</u>
$gnum$ - Numerical estimation of function Gradient. Computation will use the returned function value from "f1" or "f2" defined in task 3.
$Hnum$ - Numerical estimation of function Hessian. Computation will use the returned analytical gradient value from "f1" or "f2"defined in task 3.

Important note: don't compute unnecessary things! (See task 3)

3 COMPARISON AND CONCLUSIONS

We should be able to check whether our analytical differentiation is correct and one way to do so is to calculate the numerical differentiation and compare between the two. In this part we will create Matlab script for comparison and check accuracy.

Task 5

Create a Matlab script "main" which computes the difference between the analytical and the numerical differentiations. For each one of “f1” and “f2” the script will calculate the analytical solution using "f1"/”f2” and numerical estimation using "numdiff".
The script will plot the following graphs for each of the functions from tasks 1 and 2:

1. Difference between numerical an analytical gradient, you should get a vector of gradient length, just use plot.
2. Difference between numerical an analytical Hessian, you should get a matrix, you can use imagesc to plot it.
3. Calculate infinity norm (maximal element of gradient vector) of the error between analytical and numerical gradient for several values of epsilon (increment of x) and plot the resulting graph(use logarithmic scale), find for which epsilon value we get the minimal difference between analytical and numerical computation.
4. Calculate infinity norm (maximal element of Hessian matrix) of the error between analytical and numerical Hessian for several values of epsilon (increment of x) and plot the resulting graph (use logarithmic scale), find for which epsilon value we get the minimal difference between analytical and numerical computation.

Conclusions

You are welcome to search Wikipedia on Numerical Differentiation to infer the desired accuracy and verify your Matlab functions are correct. Also, you can search Wikipedia on Automatic differentiation which allows you to automatically compute the Gradient and the Hessian only from the function values and see if you can implement it.