Seung Jun Baek

1 Outline

In this assignment, you are asked to implement gradient descent algorithm with backtracking line search.

2 Specification

In this assignment, you are asked to write a function that implements gradient descent with backtracking line search. Your function must have the format:

- function name: `min_gd(fun,x0,grad,args=())`
- parameters:
  - `fun` : The objective function $f : \mathbb{R}^n \to \mathbb{R}$ to be minimized. You can invoke the function by `fun(x, *args)` which should return `float`. `x` is an 1-D array with shape (n,) and `args` is a tuple of the fixed parameters needed to completely specify the function.
  - `x0` : This is `ndarray` with shape $(n,)$, and represents the n dimensional vector specifying the initial point.
  - `grad`: Gradient $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$ of function `fun`. should return 1-D array with shape $(n,)$ of type `float`. The gradient can be evaluated by calling `grad(x,*args)` – it shares the same arguments as `fun`.
  - `args`: *tuple, optional.* Extra arguments passed to the objective function and its derivatives (`fun`, and `grad` functions).

Also meet the following requirements

- All the variable names are case sensitive.
- Use the following parameters for backtracking line search:

$$\alpha = 0.3,\ \beta = 0.8$$

  Or you can choose your own parameter if you want.

- Stopping criteria: there are several possibilities, but you can consider

$$\|\nabla f\| \leq \epsilon$$

  for some appropriate small value of tolerance ϵ.

3 What to submit

- Submit a python module implementing function `min_gd`; the filename of the module must be `gd.py`.
- Upload your `gd.py` file at Blackboard before deadline (no late submission accepted).

Your `gd.py` file look something like

```
import numpy as np

def min_gd(fun,x0,grad,args=()):

# your code goes here
```

If necessary you can `import` other libraries, or you can define other additional functions of your own in your `gd.py` file.

4 How to test your module

In the blackboard, I have uploaded `hw.py` so that you can test your module. The file starts with the following lines:

```
import numpy as np
from scipy.optimize import minimize
import gd
```

You can put your `gd.py` as the same directory as `hw.py` file while you test-run `hw.py`. `hw.py` will use `minimize` function from `scipy` library which implements a generic minimization. Documents on python `minimize` can be found at `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html`.

The following explains what `hw.py` do. We try to minimize the following function

$$f(x) = \|Ax - b\|_2^2$$

which is defined as follows:

```
# Least Squares function
def LeastSquares(x,A,b):
    return np.linalg.norm(A@x-b)**2
```

Also its gradient

$$\nabla f(x) = 2(A^T Ax - A^T b)$$

```
# gradient
def grad_LeastSquares(x,A,b):
    return 2*((A.T@A)@x-A.T@b)
```

Here A and b are parameters to be passed to `minimize` and `min_gd` functions. They are randomly generated according to Gaussian distribution using `np.random.normal`.

Firstly the function is minimized by using `minimize` function

```
res=minimize(fun=LeastSquares,x0=x0,args=(A,b))
```

See how `LeastSquares` is passed to the function along with A and b. The solution from `minimize`, which can be accessed by `res.x`, can be used as reference. Then you run your function

```
x=gd.min_gd(fun=LeastSquares,x0=x0,grad=grad_LeastSquares,args=(A,b))
```

Similarly note how `LeastSquares` and `grad_LeastSquares` and A, b are passed to `min_gd`. Finally your solution and the solution from `minimize` is compared by

```
# show error between built-in and your solutions
print('error :',np.linalg.norm(x-res.x))
```

The following is captured from python console output in case of a successful run of `hw.py`.

```
solution from minimize: message:  Optimization terminated successfully.
solution from minimize: success: True
solution from minimize: solution x: [-0.06863077  0.17205247 -0.10610076
 0.2227344  -0.04588808 -0.03826752
  0.0250977  -0.00769958 -0.08309249  0.0087596 ]
solution from min_gd: [-0.06863075  0.17205249 -0.10610079  0.22273441
-0.04588809 -0.03826752
  0.02509768 -0.00769958 -0.08309254  0.0087596 ]
error : 7.169960727328429e-08
```

In particular, focus on the following message output from the capture

- `solution from minimize: message: Optimization terminated successfully.` At your test run, you should see this message too ; otherwise something has gone wrong.
- `solution from minimize: success: True` At your test run, you should see this message too ; otherwise something has gone wrong.
- `error : 7.169960727328429e-08`. You should also see a small value in this error output. If this is big, something has gone wrong.

5 Grading

- 5 points if your module works correctly. Specifically the error between your solution and `minimize` function is more than 10^{-3} for randomly generated A and b.

The rest of case is 0 points, i.e., if you do not submit (or late), or if your file does not compile correctly, or produces wrong results.