Project 1: Steepest Descent Method

January 9, 2019

1 Project: Steepest Descent Method

In this project, you will write program to solve some optimization problems with the `Steepest Descent Method`, it is also called `Gradient Descent Method`.

1.1 Guideline

Here is a guideline for your project.

- The project is designed to help understand the topic, so please do not copy and paste.
- You are welcome to work in groups (no more than 3 in one group), but you should write it up for your submission/exercises on your own.
- The standard submission of project contains the following documents:
  - The code files
  - A `README` file
  - A brief writeup document for the project

In your `README` file, there are several things you need to declare:

- What language are you using to write the program;
- How to run the code;
- Contributors.

You can add additional information such as possible issues, future plan in the `README` file as well. For the writeup document, you will be answering the questions in the project.

1.2 Project Description

The `Steepest Descent Method` iteratively solves the step length (which is a one-dimensional optimization problem) using the steepest direction at each iteration. In practice, it is not necessary to find the exact optimal solution during each iteration. In our project, we will try to implement several strategies for this problem.

1.2.1 Task 1: Exact Steepest Descent Method

In this task, you will implement the exact steepest descent method as the following algorithm. The standard (exact) algorithm is as the following:

1. Let $k \leftarrow 0$, pick a starting point $\hat{x}$ and let $x_k \leftarrow \hat{x}$
2. Find $p_k \leftarrow -\nabla f(x_k)$, if $\|p_k\| < \delta$, then return x_k, otherwise solve $\alpha_k \leftarrow \min_{\alpha>0} f(x_k + \alpha p_k)$
3. $x_{k+1} \leftarrow x_k + \alpha_k p_k$, $k \leftarrow k+1$, go to step 2.

For the 1D minimization problem at step 2, you can use the Newton's method to solve the local minimum. If you use Python, you can use

```
from scipy.optimize import minimize_scalar
```

to import the `minimize_scalar` function to solve the 1D optimization problem. For MATLAB, you can use the function `fminbnd` for the same purpose.

Task: You will be using your algorithm to solve the unconstrained optimization problem $\min_{x=(x_1,x_2)} f(x)$ for the objective function f as the following quadratic function:

$$f(x_1, x_2) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$$

The solution of above optimization problem is $(1, 3)$. For your implementation, you will use starting point from $\hat{x} = (0, 0)$ and the tolerance $\delta = 10^{-9}$.

1.2.2 Task 2: Inexact Steepest Descent Method (fixed step length)

It is known that we could use a fixed step length $\alpha_k = \alpha^*$ for all iteration as long as α^* is small enough. The algorithm is stated in the following:

The fixed step length steepest descent algorithm is as the following:

1. Let $k \leftarrow 0$, pick a starting point $\hat{x}$ and let $x_k \leftarrow \hat{x}$ and select α^* small enough;
2. Find $p_k \leftarrow -\nabla f(x_k)$, if $\|p_k\| < \delta$, then return x_k, otherwise solve $x_{k+1} \leftarrow x_k + \alpha^* p_k$, $k \leftarrow k+1$, go to step 2.

As a remark: For the neural network optimization problem, this α^* is often called learning rate.

Task: You will be using your algorithm to solve the unconstrained optimization problem $\min_{x=(x_1,x_2)} f(x)$ for the objective function f as the following quadratic function:

$$f(x_1, x_2) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$$

The solution of above optimization problem is $(1, 3)$. For your implementation, you will use starting point from $\hat{x} = (0, 0)$ and the tolerance $\delta = 10^{-9}$ and $\alpha^* = 10^{-3}$.

1.2.3 Task 3: Experiment on Rosenbrock function

Use above two algorithms on the following Rosenbrock function:

$$f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$$

You will try the following starting points (other parameters are fixed as the previous tasks) and answer the questions below:

1. starting point $\hat{x} = (1.2, 1.2)$, find out how many iterations have the above methods used, what is the convergence rate?
2. starting point $\hat{x} = (-1.2, 1)$, find out how many iterations have the above methods used, what is the convergence rate?

2 Starter code kit

The starter code kit is in Python (MATLAB users should have no difficulty to understand.) It is up to you to following the starter code kit's style or not. As a reminder, the kit only fills out framework, there are some details you will have to write on your own.

3 Submission

If you prefer to use the GitHub or Bitbucket, etc. to store your code, then it is OK to simply submit the repository's link on a text file (if your repository is public), otherwise, please submit all files in a zip file through Canvas.