Project 1

Wenxuan Hong 82393226 Tianyun Yuan 86723015

February 2023

1 Secant Line Search

Line search is the algorithm that finds or approximates the minimizer of a function on a given direction. In other words, it is a sub-problem where instead of finding the minimizer to an unconstrained problem over R^n, one finds the minimizer to a constrained problem on span$\{\mathbf{d}\}$. But how do we solve this constrained problem? We will need some kind of line search algorithm.

Given a function $f(x)$ and a direction $\mathbf{d}$, Secant Method can efficiently approximate the minimizer on this direction using function values $f(x)$ and first order derivatives $f'(x)$.

Secant line search method is derived from Newton's line search method, which has the form

$$x_{k+1} = x_k - \frac{f^{'}(x_k)}{f^{''}(x_k)}$$

Instead of approximating the function itself, Newton's method approximates the Taylor expansion of the original function to the second degree and uses FONC to derive that the derivative of approximated function equals zero. By solving the equation we get the formula above. Newton's method converges very quickly, but the main drawback of this method is the need to compute first and second-order derivatives for each step, which can be expensive or unfeasible. Hence we use secant method instead. Consider the second order derivative $f^{''}(x)$ at x_k, it is approximately

$$f^{''}(x_k) \approx \frac{f^{'}(x_k) - f^{'}(x_{k-1})}{x_k - x_{k-1}}$$

We plug in this approximation into the Newton's line search method and get this

$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f^{'}(x_k) - f^{'}(x_{k-1})} f^{'}(x_k)$$

This is the algorithm we implemented in the function **lineSearchSecant(grad,x,d)**

```
function [x_new] = linesearchSecant(grad,x,d)
% This function computes the line search on direction d with secant
     method
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Input
% grad: gradient function
% x: starting point
% d: gradient descent direction
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Output
% x_new: minimizer of the function on the descent direction
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
xa = x; xb = xa + d'/norm(d); d_norm = norm(d); % Initialization
for i = 1:100
    fda = d'*(grad(xa)*d)/d_norm; % Function gradient of xa
    projected on the descent direction
    fdb = d'*(grad(xb)*d)/d_norm; % Function gradient of xb
    projected on the descent direction
    alpha = norm(xb-xa)/(norm(fdb)-norm(fda)); x_new = xb - alpha*
    fdb; % Compute the x of the next step
    xa = xb; xb = x_new; % Update virable values
    if abs(grad(x_new)*d) <= 0.01* abs(grad(x)*d)
        break
    end
end
end
```

2 Gradient Descent

Gradient Decent is an algorithm that solves the unconstrained minimizing problem over R^n by deconstructing it to sub-problems where, in each problem, it finds the minimizer to a constrained problem on span$\{\mathbf{d}\}$. More specifically, it sets the direction $\mathbf{d}$ to be the gradient of function $f^{'}(x)$ at x_k.

It has the form

$$x_{k+1} = x_k - \alpha f^{'}(x)$$

where, specifically, α is $\frac{x_k - x_{k-1}}{f^{'}(x_k) - f^{'}(x_{k-1})}$, as shown in the last section.However, since we have already defined the function **lineSearchSecant(grad,x,d)** in the last section, we need only to call it to do the line search for us when needed.

This is the algorithm we implemented in the function **GD(grad,x,ep)**

```
function [x] = GD(grad,x,ep)
% This function computes the gradient descent with secant method
    for 5000
% steps or untill the gradient decreases to ep
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Input
% grad: gradient function
% x: starting location x
% ep: termination thereshold of gradient
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Output
% x: minimizer according to secant method
for i = 1:5000
    d = -grad(x)'; % update d to the negative of gradient vector
    x = linesearchSecant(grad,x,d); % compute the minimizer of line
     search on direction d
    if norm(d) <= ep
        break
    end
end
end
```

3 Other functions

Because the secant method only requies the values of x and first order derivatives, we need the following two functions to compute the derivatives needed in secant line search

```
function [fd] = func_deriv(x)
fd = zeros(1,3);
fd(1) = 4*(x(1)-4)^3;
fd(2) = 2*(x(2)-3);
fd(3) = 16*(x(3)+5)^3;
end
```

```
function [fd] = Rosenbrock_grad(x)
fd = zeros(1,2);
fd(1) = 400*x(1)^3 + (2-400*x(2))*x(1)-2;
fd(2) = 200*(x(2)-x(1)^2);
end
```

4 main code

With all these functions defined, the main code will be fairly brief, consisting of only us calling the GD functions and displaying results.

```
clear, clc
x = [4,5,1];grad = @func_deriv;ep = 1e-6;
x_star = GD(grad,x,ep);
disp(func(x_star))
```

```
    2.3797e-10
```

```
x = [-2,2];grad = @Rosenbrock_grad; ep = 1e-4;
x_star = GD(grad,x,ep);
disp(x_star)
```

```
    0.9999    0.9998
```

5 Appendix: Link to Full Code

https://github.com/YS0meone/Math110A$_{P}rojects$