전기전자공학수학

Computer Simulation HW1

2018142023 조성민

7.11 Write a MATLAB function that implements a line search algorithm using the secant method. The arguments to this function are the name of the M-file for the gradient, the current point, and the search direction. For example, the function may be called linesearch_secant and be used by the function call alpha=linesearch_secant('grad',x,d), where grad.m is the M-file containing the gradient, x is the starting line search point, d is the search direction, and alpha is the value returned by the function [which we use in the following chapters as the step size for iterative algorithms (see, e.g., Exercises 8.25 and 10.11)].

Note: In the solutions manual, we used the stopping criterion $|\boldsymbol{d}^\top \nabla f(\boldsymbol{x} + \alpha \boldsymbol{d})| \geq \varepsilon |\boldsymbol{d}^\top \nabla f(\boldsymbol{x})|$, where $\varepsilon > 0$ is a prespecified number, ∇f is the gradient, $\boldsymbol{x}$ is the starting line search point, and $\boldsymbol{d}$ is the search direction. The rationale for the stopping criterion above is that we want to reduce the directional derivative of f in the direction $\boldsymbol{d}$ by the specified fraction ε. We used a value of $\varepsilon = 10^{-4}$ and initial conditions of 0 and 0.001.

<Theory>

Line Search in Multidimensional Optimization

-Vector space에서 d방향으로 탐색한다.

-Iterative algorithm: $x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}$ for minimizing $\phi_k(\alpha) = g(x^{(k)} + \alpha_k d^{(k)})$

-Initial point: $x^{(0)}$

-Search direction: $d^{(k)}$

-Step size: α_k

-Derivative: $\phi'_k(\alpha) = d^{(k)T} \nabla g(x^{(k)} + \alpha d^{(k)})$

<Implementation>

-$g(x_1, x_2, x_3) = (x_1 - 4)^4 + (x_2 - 3)^2 + 4(x_3 + 5)^4$

-$\nabla g = 4(x_1 - 4)^3 + 2(x_2 - 3) + 16(x_3 + 5)^3$

-$\phi'_k(\alpha) = d^{(k)T} \nabla g(x^{(k)} + \alpha d^{(k)})$

-$\alpha^{(k+1)} = \dfrac{\phi'_k(\alpha^{(k)})\alpha^{(k-1)} - \phi'_k(\alpha^{(k-1)})\alpha^{(k)}}{\phi'_k(\alpha^{(k)}) - \phi'_k(\alpha^{(k-1)})}$

<MATLAB code>

1. linesearch_secant

```
function alpha=linesearch_secant(grad,x,d)

epsilon=10^(-4);
max=100;
alpha_curr=0;
alpha=0.001;
dphi_zero=feval(grad,x)'*d;
dphi_curr=dphi_zero;

i=0;
while abs(dphi_curr)>epsilon*abs(dphi_zero)
    alpha_old=alpha_curr;
    alpha_curr=alpha;
    dphi_old=dphi_curr;
    dphi_curr=feval(grad,x+alpha_curr*d)'*d;
    alpha=(dphi_curr*alpha_old-dphi_old*alpha_curr)/(dphi_curr-dphi_old);
    i=i+1;
    if (i >= max) & (abs(dphi_curr)>epsilon*abs(dphi_zero))
        disp('Line search terminating with number of iterations:');
        disp(i);
        break;
    end
end
```

-Line search tolerance: $\varepsilon = 10^{-4}$

-Maximum number of iterations: 100

-alpha_curr=0

-alpha=0.001

-alpha_old=$\alpha^{(k-1)}$

-alpha_curr=$\alpha^{(k)}$

-dphi_old=$\phi'_k(\alpha^{(k-1)})$

-dphi_curr=$\phi'_k(\alpha^{(k)})$

2. grad(g)

```
function y=g(x)
    y=[4*(x(1)-4).^3; 2*(x(2)-3); 16*(x(3)+5).^3];
end
```

-$\nabla g = 4(x_1-4)^3 + 2(x_2-3) + 16(x_3+5)^3$

<Result & Analysis>

```
>> linesearch_secant(@g,[4;2;-1],[0;-2;1024])

ans =

   -0.0038
```

-linesearch_secant 함수에 gradient g, starting line search point $[4,2,-1]^T$, search direction $[0,-2,1024]^T$을 입력했을 때 alpha값이 -0.0038 임을 알 수 있다.