Problem 1: Proving what Newton method is up to

Problem Statement

Newton's method minimizes the local quadratic approximation of a function, if its Hessian at the current iterate is positive definite. We showed that Newton's method indeed finds a *local* minimizer of the quadratic approximation. Here, we show that it finds the *global* minimizer through the following steps.

a

If Q is any PD matrix, show that:

$$\frac{1}{2}x^TQx + c^Tx = \frac{1}{2}(x + Q^{-1}c)^TQ(x + Q^{-1}c) - \frac{1}{2}c^TQ^{-1}c$$

b

Using the prior result, argue that the function $\frac{1}{2}x^TQx + c^Tx$ is minimized *globally* at $x^* = -Q^{-1}c$.

c

Recall that for any function $f : \mathbb{R} \to \mathbb{R}$, its local quadratic approximation at x^k is given by:

$$f^q(x) \coloneqq f(x^k) + [\nabla f(x^k)]^T(x - x^k) + \frac{1}{2}(x - x^k)^TH(x^k)(x - x^k)$$

Assume the Hessian $H(x^k)$ is positive definite. Utilize the prior result to conclude that the *global* minimizer of f^q is given by $x^k - [H(x^k)]^{-1}\nabla f(x^k)$. Notice that the minizer is indeed x^{k+1}, as defined by Newton's method.

Solution

a

Simply expand the right side:

$$= \frac{1}{2}(x^TQ(x + Q^{-1}c) + c^T(Q^{-1})^TQ(x + Q^{-1}c)) - \frac{1}{2}c^TQ^{-1}c$$

$$= \frac{1}{2}(x^TQx + x^TQQ^{-1}c + c^T(Q^{-1})^TQx + c^T(Q^{-1})^TQQ^{-1}c) - \frac{1}{2}c^TQ^{-1}c$$

Noting that (per the instructor comment on Piazza) $Q^T = Q$ and that $Q^{-1} \succ 0$:

$$= \frac{1}{2}(x^TQx + x^Tc + c^Tx + c^TQ^{-1}c) - \frac{1}{2}c^TQ^{-1}c$$

Finally, note that $c^Tx = x^Tc \implies c^Tx + x^Tc = 2c^Tx$ and thus:

$$= \frac{1}{2}x^TQx + c^Tx + \frac{1}{2}c^TQ^{-1}c - \frac{1}{2}c^TQ^{-1}c = \frac{1}{2}x^TQx + c^Tx$$

□

b

By definition, if $Q \succ 0$, $x^TQx > 0 \forall x \neq 0$ and $x^TQx = 0 \iff x = 0$. As such, $\min_x x^TQx = 0$ and $\arg\min_x x^TQx = 0$. Given the prior result, if we substitute in $x := x^* = -Q^{-1}c$ we find:

$$\frac{1}{2}x^TQx + c^Tx = \frac{1}{2}(x + Q^{-1}c)^TQ(x + Q^{-1}c) - \frac{1}{2}c^TQ^{-1}c$$

$$= \frac{1}{2}(-Q^{-1}c + Q^{-1}c)^TQ(-Q^{-1}c + Q^{-1}c) - \frac{1}{2}c^TQ^{-1}c$$

$$= \frac{1}{2}(0)^TQ(0) - \frac{1}{2}c^TQ^{-1}c = -\frac{1}{2}c^TQ^{-1}c$$

which is a constant. Note that the term that was dropped cannot be negative. Thus, this result must be the global minimum of the function. □

c

First, we need to note that $f(x^k)$ is a constant, as x^k is a parameter and will not change as a function of x. Furthermore:

$$\arg\min_x g(x) = \arg\min_x g(x) + k$$

i.e. addition of a constant does not change the location of the optimum.

So: simply let

$$c' := \nabla f,\ x' = x - x^k, Q' = H$$

We can then see it matches the form:

$$f(x^k) + \frac{1}{2}x'^TQ'x' + c'^Tx'$$

Given our expression for x', we can see

$$x'^* = x^* - x^k = x^k - H^{-1}\nabla f - x^k = -H^{-1}\nabla f = -Q'^{-1}c'$$

which is identical to the result in the prior section.[1] □

[1] Kudos to Will V. for pointing out the expression should be massaged to match the prior expression from the get-go. I had previously spent a few hours chasing my tail trying to make the expressions match *after* I substituted the value for the optimum. This was fruitless.

Problem 2: Newton's method needs a touch-up

Problem Statement

In Newton's method, if the Hessian at the current iterate $H(x^k)$ is not PD, then $-[H(x^k)]^{-1}\nabla f(x^k)$ may not be a descent directon. Then, we modify the Hessian to $H(x^k)+D^k$ where D^k is a diagonal matrix with nonnegative diagonal entries. Let us design D^k to ensure that $H(x^k)+D^k$ is PD. The following corollary of Gershgorin's circle theorem will prove useful.

Theorem 0.1. *If λ is any eigenvalue of an arbitrary matrix $A \in \mathbb{R}^{n\times n}$, then*

$$|\lambda - Aii| \leq \sum_{j\neq i} |A_{ij}|$$

for some $i = 1, \ldots, n$.

a

Using this theorem, find a sufficient condition on the diagonal entries of A s.t. all eigenvalues of A are positive.

b

Using the prior result, find a diagonal matrix D^k s.t. that all eigenvalues of $H(x^k) + D^k$ are nonnegative, and all diagonal entries of D^k are nonnegative.

c

If any eigenvalue of $H(k) + D^k$ is close to zero but positive, then it is close to being singular and its inverse is susceptible to noise. Modify your answer in the prior section to ensure that all the eigenvalues are greater than $\frac{1}{2}$.

d

The file applyNewtonMethod.m implements a basic newton method to the function

$$f(x_1, x_2) := \cos(x_1^2 - 2x_2) + \sin(x_1^2 + x_2^2)$$

starting from $x^0 = (1.2, 0.5)$. The program also draws the contour plot and the surface plot of f.

1. Verify the Hessian at the starting point is NOT PD.
2. Does the Newton method converge? If yes, does it converge to a local minimizer of f?
3. Fill in the missing code in modifyHessian.m that takes $H(x^k)$ as input and gives $H(x^k)+D^k$ as output. Utilize your condition in part (c) to design D^k. Submit your code. Using your code, compute $H(x^0) + D^0$; i.e the modified Hessian at the starting point.
4. Uncomment the relevant lines in applyNewtonMethod.m to run the modified Newton method. Report if the algorithm converges to a local minimizer of f.

Solution

a

Using the geometric interpretation (borrowing heavily from the Wikipedia page[2]), we can note that the theorem essentially describes a bounded estimate of an eigenvalue, where the center of circle is located at the value of the diagonal element in a row (A_{ii}) and its radius is the sum of the magnitude of all other elements in the row ($\sum_{i \neq j} A_{ij}$). In order to bound the eigenvalue to be positive, this circle must exist solely within the RHS of the complex plane. Thus: Given a square matrix $A \in \mathbb{R}^{n \times n}$, a sufficient (albeit not necessary!) condition is:

$$A_{ii} > \sum_{i \neq j} |A_{ij}| \forall i \in 1, \ldots, n$$

In plain english: we have to move the bounding circle via a translation of its center (the diagonal element of that row) to be larger than than he radius. If we only require non-negative eigenvalues, the relationship could be relaxed to be "greater than or equal to".

b

$$D_{ii} + A_{ii} > \sum_{i \neq j} |A_{ij}| \implies D_{ii} > \sum_{i \neq j} |A_{ij}| - A_{ii}$$

c

$$D_{ii} > \sum_{i \neq j} |A_{ij}| - A_{ii} + \frac{1}{2}$$

d

Note: I used GNU Octave[3] instead of MATLAB because I'm too lazy to get it installed and working on my machine (plus I'm not a fan of closed-source proprietary languages).

1. I checked `all(eig(Hk)>0)` for iterate 0, which reports `false`. The matrix is not PD.
2. It does after 4 iterations. It is not a local minimum; again, the Hessian is evaluated to be not PSD. The necessary second-order optimality condition does not hold: it is not a local minimum.
3. See code listing at end of the document.

$$\tilde{H}^0 = \begin{bmatrix} 5.3041 & 4.8041 \\ 4.8041 & 5.3041 \end{bmatrix}$$

4. It does. Script output at final iteration:

```
Iteration #17
Current values of (x,y) = [27.4427      -28.5679]
Current norm of gradient =1.2935e-07
Function value at last iterate =-2
Iterate is a local minimum: True
```

[2]Wikipedia: Gershgorin's Circle Theorem Example (click me!)
[3]GNU Octave (click me!)

Code

```

function HTilde = modifyHessian(H)
    n = size(H, 1);

    % Problem 4.1: Write your code here to compute D
    % using your answer in part (c).
    h_ii = diag(H);
    D_ii = zeros(size(h_ii));
    off_diag = H-diag(h_ii);

    for i = 1:n
        r = sum(abs(off_diag(i,:)));
        min_eigenvalue_estimate = h_ii(i) - r;
        if min_eigenvalue_estimate < (1/2)
            D_ii(i) = 1/2 + r - h_ii(i);
        end
    end

    D = diag(D_ii);
    HTilde = H + D;
end
```

```
% Code to run a generic Newton method.
clear all
close all
clc; history -c

MAX_ITER = 100;

% Function and its gradient and hessian.

f = @(x, y) (cos(x.^2 - 3*y)  + sin(x.^2 + y.^2));

gradf = @(x, y) ([ 2*x*cos(x^2 + y^2) - 2*x*sin(x^2 - 3*y); ...
                   3*sin(x^2 - 3*y) + 2*y*cos(x^2 + y^2) ...
                 ]);

hessf = @(x,y) [ 2*cos(x^2 + y^2) - 2*sin(x^2 - 3*y) - 4*x^2*cos(x^2 - 3*y) - 4*x^2*sin(x^2 + y^2), ...
                   6*x*cos(x^2 - 3*y) - 4*x*y*sin(x^2 + y^2); ...
                 6*x*cos(x^2 - 3*y) - 4*x*y*sin(x^2 + y^2), ...
                   2*cos(x^2 + y^2) - 9*cos(x^2 - 3*y) - 4*y^2*sin(x^2 + y^2)];

% Plot the function and its contour plot to
% appreciate how this function looks like
[Xgrid, Ygrid] = meshgrid(-1.5:0.1:1.5, -1.5:0.1:1.5);
Z = f(Xgrid, Ygrid);

subplot(2,1,1), contour(Xgrid, Ygrid, Z, 40,'Linewidth',2),
grid on, xlabel('$x_1$', 'Interpreter','Latex', 'Fontsize', 20),
ylabel('$x_2$', 'Interpreter','Latex', 'Fontsize', 20),

subplot(2,1,2), surf(Xgrid, Ygrid, Z), grid on,
xlabel('$x_1$', 'Interpreter','Latex', 'Fontsize', 20),
ylabel('$x_2$', 'Interpreter','Latex', 'Fontsize', 20),
zlabel('$f(x_1, x_2)$', 'Interpreter','Latex', 'Fontsize', 20),



% Initialize at (1.2, 0.5).
xk = [1.2; 0.5];

% Tolerance for the gradient value.
tolerance = 1e-6;

% Variables required for the iteration.
shouldIterate = true;
iterationK = 1;
errorGF = 0
while (shouldIterate)

    % Display the current iteration.
    display(strcat('Iteration # ', num2str(iterationK)))
    display(strcat( ...
        'Current values of (x,y) = [', ...
        num2str(xk'), ...
        ']'...
    ))
    % Compute the norm of the gradient.
    errorGF = norm(gradf(xk(1), xk(2)), 2);

    display(strcat( ...
        'Current norm of gradient = ', ...
        num2str(errorGF) ...
    ))
    % Added by ESilk to display if the Hessian is PD!
    if(iterationK == 1)
        Hk = hessf(xk(1), xk(2));
        output_str = {"False", "True"};
```

```
        H_is_PD = "False";
        if(all(eig(Hk)>0))
          H_is_PD = "True";
        end
        printf('H > 0: %s', H_is_PD)
        disp('')
    end
    %pause
        % Uncomment the previous line if you want to
        % look at each new iterate.

    if (errorGF > tolerance)

        % Compute the current Hessian
        Hk = hessf(xk(1), xk(2));
        Hk = modifyHessian(Hk);
        if(iterationK == 1)
            display('Modified H0:')
            display(num2str(Hk))
        end
        display('')
        assert (all(eig(Hk) > 0))
            % Problem 4.1: Uncomment the two lines above and include your function
            % to modify the Hessian as per your answer to part (b).


        % Compute the Newton direction.
        sk = - inv(Hk) * gradf(xk(1), xk(2));

        alphak = 1;

        % alphak = lineSearch(f, gradf, xk, sk, 0.0001);
        % if alphak == -1
        %    break
        % end
            % Probelm 4.2: Uncomment the above lines for implementing your own
            % line search function. It also enforces that the line
            % search in fact converges.

        % Compute the next iterate.
        xk = xk + alphak * sk;

        iterationK = iterationK + 1;
    else
        % Error is within tolerance
        shouldIterate = false;
    end

    if iterationK == MAX_ITER
        display('Did not converge within 100 iterations')
        break
    end
end

display(strcat( ...
    'Function value at last iterate = ', ...
    num2str(f(xk(1), xk(2))) ...
))
Hk = hessf(xk(1), xk(2));

H_is_PD = "False";
if(all(eig(Hk)>=0))
  H_is_PD = "True";
end

display(strcat('Iterate is a local minimum:', ...
  H_is_PD
 ))
```