Project 2 Report

CSS322 Scientific Computing
Sirindhorn International Institute of Technology, Thammasat University
Semester 1 Academic Year 2021

Paphana Yiwsiw
6222780379
Section 3

December 5, 2021

Contents

1 Newton's method for unconstrained optimization

The following is the result from running newton's method for unconstrained optimization on $f(x) = sin(x_1) - cos(x_2)$. The value of $x^{(k)}$ and the corresponding $f(x^{(k)})$ at $k = 0, 5, 10, 15$.

- $k = 0, x^{(0)} = \begin{bmatrix} 0.5434 \\ 0.2784 \end{bmatrix}, f(x^{(0)}) = -0.4445.$
- $k = 5, x^{(5)} = \begin{bmatrix} 1.5708 \\ 0 \end{bmatrix}, f(x^{(5)}) = 0.$
- $k = 10, x^{(10)} = \begin{bmatrix} 1.5708 \\ 0 \end{bmatrix}, f(x^{(10)}) = 0.$
- $k = 15, x^{(15)} = \begin{bmatrix} 1.5708 \\ 0 \end{bmatrix}, f(x^{(15)}) = 0.$

2 Naive Random Search Algorithm

The following is the result from running naive random search algorithm on perturbed $f(x)$. The value of $x^{(k)}$ before picking candidate and corresponding value $Z(x^{(k)})$ at $k = 0, 25, 50, 75, 100$.

- $k = 0, x^{(0)} = \begin{bmatrix} 583 \\ 556 \end{bmatrix}, Z(x^{(0)}) = -0.9699.$
- $k = 25, x^{(25)} = \begin{bmatrix} 585 \\ 555 \end{bmatrix}, Z(x^{(25)}) = -1.5657.$
- $k = 50, x^{(50)} = \begin{bmatrix} 585 \\ 555 \end{bmatrix}, Z(x^{(50)}) = -1.5657.$
- $k = 75, x^{(75)} = \begin{bmatrix} 585 \\ 555 \end{bmatrix}, Z(x^{(75)}) = -1.5657.$
- $k = 100, x^{(100)} = \begin{bmatrix} 585 \\ 555 \end{bmatrix}, Z(x^{(100)}) = -1.5657.$

From observation, it seemed that it is currently stuck in a region around local optimizer, when neighborhood have larger objective function values than the current point.

3 Simulated Annealing Algorithm

The following is the result from running simulated annealing algorithm applied on perturbed $f(x)$. The value of $x^{(k)}$ before picking candidate and corresponding value $Z(x^{(k)})$ at $k = 0, 25, 50, 75, 100$.

- $k = 0, x^{(0)} = \begin{bmatrix} 40 \\ 44 \end{bmatrix}, Z(x^{(0)}) = 0.0648.$
- $k = 25, x^{(25)} = \begin{bmatrix} 37 \\ 58 \end{bmatrix}, Z(x^{(25)}) = -0.3207.$
- $k = 50, x^{(50)} = \begin{bmatrix} 32 \\ 65 \end{bmatrix}, Z(x^{(50)}) = 0.2214.$
- $k = 75, x^{(75)} = \begin{bmatrix} 40 \\ 41 \end{bmatrix}, Z(x^{(75)}) = -0.0050.$
- $k = 100, x^{(100)} = \begin{bmatrix} 47 \\ 39 \end{bmatrix}, Z(x^{(100)}) = 0.0570.$

Noted that the best-so-far point from running simulated annealing algorithm is at $x = \begin{bmatrix} 49 \\ 39 \end{bmatrix}$ and $Z(x) = -0.5004$.

MATLAB Code

Newton's Method for Unconstrained Optimization

```
% CSS322 Project 2: Optimization
% Paphana Yiwsiw 6222780379
% Part I: Newton's Method
%  (Newton's method for uncontraint optimization)

function [xk,fxk] = newtons()
    % set random seed
    s = rng;
    rng(100);
    % Random initial value
    xk = rand(2,1);
    fxk = f(xk);
    gfxk = gradientf(xk);
    hfxk = hessianf(xk);
    % Newton's method implementation
    for k = 0:15
        % solve linear system of equation for h(k)
        hk = linearsolve(hfxk,-gfxk);
        % calculate x(k+1)
        xkp1 = xk + hk;
        fxkp1 = f(xkp1);
        % report value
        if mod(k,5) == 0
            fprintf("\nk = %d\n",k);
            fprintf("x(k)  = [ %.4f ; %.4f ]\n",xk);
            fprintf("f(x(k))  = %.4f\n",fxk);
            fprintf("x(k+1)  = [ %.4f ; %.4f ]\n",xkp1);
            fprintf("f(x(k+1))  = %.4f\n",fxkp1);
        end
        % update xk, gradient, and hessian matrix
        xk = xkp1;
        fxk = fxkp1;
        hfxk = hessianf(xkp1);
        gfxk = gradientf(xkp1);
    end
    % end session
    rng(s);
end

% given function f(x)
function [fx] = f(x)
    fx = sin(x(1))-cos(x(2));
end

% gradient of fx
function [gfx] = gradientf(x)
    gfx = [cos(x(1));sin(x(2))];
end

% hessian matrix of fx
function [hfx] = hessianf(x)
    hfx = [-sin(x(1)), 0; 0, cos(x(2))];
end

```

```
function [x] = linearsolve(A,b)
    % PTLU factorization
    [L,U,P] = lu(A);
    % b_hat = P*b
    b_hat = P*b;
    % L*w=b_hat by forward substitution
    w = forwardsub(L,b_hat);
    % U*x=w by backward substitution
    x = backwardsub(U,w);
end

function w = forwardsub(L,b)
    [n,~] = size(b);
    w = zeros(size(b));
    for j = 1:n
       w(j) = b(j)/L(j,j);
       for k = j+1:n
           b(k) = b(k) - (L(k,j)*w(j));
       end
    end
end

function x = backwardsub(U,b)
    [n,~] = size(b);
    x = zeros(size(b));
    for j = n:-1:1
        x(j) = b(j)/U(j,j);
        for k = 1:j-1
           b(k) = b(k) - (U(k,j)*x(j));
        end
    end
end
```

Naive Random Search Algorithm

```
% CSS322 Project 2: Optimization
% Paphana Yiwsiw 6222780379
% Part II: Naive random search
% Out :  xk = last point
%       fxk = value of last point

function [xk,fxk] = naive_random()

    % set random seed
    s = rng;
    rng(1000);
    % create a perturbed f(x)
    [X,Y] = meshgrid(0:0.01:2*pi,0:0.01:2*pi);
    ZZ = rand(size(X));
    Z = sin(X)-cos(Y) + ZZ;

    % Naive random search implementation
    % random initial starting point
    xk = randi(629,2,1);
    fxk = Z(xk(1),xk(2));
    [neighborX,neighborY] = meshgrid(-2:2,-2:2);
```

```
    for k = 0:100
        % pick candiddate zk from neighbor of xk
        while true
            i = randi(24);
            if i >= 13
                zk(1) = xk(1) + neighborX(i+1);
                zk(2) = xk(2) + neighborY(i+1);
            else
                zk(1) = xk(1) + neighborX(i);
                zk(2) = xk(2) + neighborY(i);
            end
            % check if out of bound or not
            if (zk(1) >= 1 && zk(1) <= 629) && ...
               (zk(2) >= 1 && zk(2) <= 629)
                break
            end
        end
        % display candidate zk
        fzk = Z(zk(1),zk(2));
        % create x(k+1) for next round
        if fzk < fxk
            xkp1 = zk;
        else
            xkp1 = xk;
        end
        fxkp1 = Z(xkp1(1),xkp1(2));
        % display when k = 0,25,50,75,100
        if mod(k,25) == 0
            fprintf("\nk = %d\n",k);
            fprintf("x(k)  = [ %d ; %d ]\nZ(x(k))  = %.4f\n",xk,fxk);
            fprintf("z(k)  = [ %d ; %d ]\nZ(z(k))  = %.4f\n",zk,fzk);
            fprintf("x(k+1)  = [ %d ; %d ]\nZ(x(k+1))  = %.4f\n",xk,fxkp1);
            fprintf("--------------------\n");
        end
        % move to next round
        xk = xkp1;
        fxk = fxkp1;
    end
    % end session
    rng(s);
end
```

Simulated Annealing Algorithm

```
% CSS322 Project 2: Optimization
% Paphana Yiwsiw 6222780379
% Part III (Extra Credit): Simulated annealing algorithm
% Out :  best_state = best-so-far point
%        best_value = value of best-so-far point

function [best_state,best_value] = simulated_annealing()
    % set random seed
    s = rng;
    rng(2000);

    % create a perturbed f(x)
```

```
    [X,Y] = meshgrid(0:0.01:2*pi,0:0.01:2*pi);
    ZZ = rand(size(X));
    Z = sin(X)-cos(Y) + ZZ;

    % simulated annealing algorithm implementation
    % random initial starting point
    xk = randi(629,2,1);
    fxk = Z(xk(1),xk(2));
    best_state = xk;
    best_value = fxk;

    [neighborX,neighborY] = meshgrid(-2:2,-2:2);
    for k = 0:100
        tk = hajek_cooling(k);
        if tk == 0
            break
        end
        % pick candiddate zk from neighbor of xk
        while true
            i = randi(24);
            if i >= 13
                zk(1) = xk(1) + neighborX(i+1);
                zk(2) = xk(2) + neighborY(i+1);
            else
                zk(1) = xk(1) + neighborX(i);
                zk(2) = xk(2) + neighborY(i);
            end
            % check if out of bound or not
            if (zk(1) >= 1 && zk(1) <= 629) && ...
               (zk(2) >= 1 && zk(2) <= 629)
                break
            end
        end
        % display candidate zk
        fzk = Z(zk(1),zk(2));
        coin_toss = rand();
        accpt_prob = acceptance_prob(tk,fzk,fxk);
        % compare coin_toss with acceptance probability
        if coin_toss < accpt_prob
            xkp1 = zk;       % HEAD
        else
            xkp1 = xk;       % TAIL
        end
        fxkp1 = Z(xkp1(1),xkp1(2));
        % display when k = 0,25,50,75,100
        if mod(k,25) == 0
            fprintf("\nk = %d\n",k);
            fprintf("tk = %.4f\n",tk);
            fprintf("x(k) = [ %d ; %d ]\nZ(x(k)) = %.4f\n",xk,fxk);
            fprintf("z(k) = [ %d ; %d ]\nZ(z(k)) = %.4f\n",zk,fzk);
            fprintf("coin toss = %.4f\n",coin_toss);
            fprintf("acceptance prob. = %.4f\n",accpt_prob);
            fprintf("x(k+1) = [ %d ; %d ]\nZ(x(k+1)) = %.4f\n",xkp1,fxkp1);
            fprintf("-------------------\n");
        end
        % move to next round
        xk = xkp1;
        fxk = fxkp1;
```

```
        % record best-so-far point
        if fxk < best_value
            best_state = xk;
            best_value = fxk;
        end
    end

    % end session
    rng(s);

    % display best value and best state
    fprintf("best_state = [ %d ; %d ]\n",best_state);
    fprintf("best_value = %.4f\n",best_value);
end

function [p] = acceptance_prob(tk,fzk,fxk)
    prob = exp(-(fzk-fxk)/tk);
    p = min([1,prob]);
end

function [tk] = hajek_cooling(k)
    gamma = 2;
    tk = gamma/log10(k+2);
end
```