Convex Optimization

Exercise 3 (100/500)

Report Delivery Date: 8 May 2018

Instructor: Athanasios P. Liavas

Student: Konidaris Vissarion 2011030123

130/1000 points

---

1. Let $a \in \mathbb{R}^n$, a point $x_0 \in \mathbb{R}^n$ and the set $S := \{x \in \mathbb{R}^n | a \leq x\}$. The projection of x_0 onto the convex set S is given by the following convex problem.

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f_0(x) = \frac{1}{2}||x_0 - x||_2^2 \\ \text{subject to} \quad & a - x \leq 0 \end{aligned}$$

We assume that the point x_* is the optimal solution of this problem. The KKT optimality conditions of the problem are:

$$\nabla f_0(x_*) + \sum_{i=1}^{n} \lambda_i \nabla f_i(x_*) = 0 \tag{1}$$

$$\lambda_i \geq 0, i = 1, 2, .., n \tag{2}$$

$$a_i - x_i \leq 0, i = 1, 2, .., n \tag{3}$$

$$\lambda_i (a_i - x_i) = 0, i = 1, 2, .., n \tag{4}$$

Solving for (1).

$$\nabla f_0(x_*) = \frac{\partial(\frac{1}{2}||x_0 - x||_2^2)}{\partial x_*} = x_* - x_0$$

$$\nabla f_i(x_*) = -e_i \in \mathbb{R}^n, i = 1, 2, ..., n$$

$$\nabla f_0(x_*) + \sum_{i=1}^{n} \lambda_i \nabla f_i(x_*) = (x_* - x_0) - \lambda = 0$$

$$\Leftrightarrow x_* = x_0 + \lambda \tag{1}$$

where e_i the standard basis vector. Combining the KKT conditions (1), (2), (3) and (4) we get:

$$\left.\begin{array}{c} \lambda_i \geq 0 \\ x_{*i} = x_{0i} + \lambda_i \geq a_i \\ \lambda_i(a_i - x_{*i}) = \lambda_i(a_i - x_{0i} - \lambda_i) = 0 \end{array}\right\}, i = 1, 2, ..., n$$

If $x_{0i} \geq a_i$ then $\lambda_i = 0$ and $x_{*i} = x_{0i}$.
If $x_{0i} < a_i$ then $\lambda_i > 0$ and $x_{*i} = a_i$.
So the projection of x_0 onto the convex set S is $P_S(x_0) = max\{a, x_0\}$.

2. Let $S := \{x \in \mathbb{R}^2 | \ ||x||_2 \leq 1, x_1 \geq 0, x_2 \geq 0, x_1 + x_2 \leq 1\}$. The set S is convex as it's the intersection of a Euclidean circle with halfspaces. We solve the problem

$$\underset{x \in S}{\text{minimize}} \quad f_0(x)$$

for two different convex cost functions f_0. Both problems are convex optimization problems.

(a) Let $f_0(x) = (x_1 - 2)^2 + (x_2 - 2)^2$. Then the problem can be rewritten as

$$\begin{array}{ll} \underset{x \in \mathbb{R}^n}{\text{minimize}} & f_0(x) = (x_1 - 2)^2 + (x_2 - 2)^2 \\ \text{subject to} & x_1^2 + x_2^2 - 1 \leq 0 \\ & -x_1 \leq 0 \\ & -x_2 \leq 0 \\ & x_1 + x_2 \leq 0 \end{array}$$

Lets calculate the gradients of the cost and constraint functions first.

$$\nabla f_0(x_*) = \begin{bmatrix} 2(x_{*1} - 2)) \\ 2(x_{*2} - 2)) \end{bmatrix},$$

$$\nabla f_1(x_*) = \begin{bmatrix} 2x_{*1} \\ 2x_{*2} \end{bmatrix}, \nabla f_2(x_*) = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \nabla f_3(x_*) = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \nabla f_4(x_*) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

The first KKT optimality condition is

$$\nabla f_0(x_*) + \lambda_1 \nabla f_1(x_*) + \lambda_2 \nabla f_2(x_*) + \lambda_3 \nabla f_3(x_*) + \lambda_4 \nabla f_4(x_*) = 0$$

$$\Leftrightarrow \begin{bmatrix} 2x_{*1} + 2\lambda_1 x_{*1} - \lambda_2 + \lambda_4 \\ 2x_{*2} + 2\lambda_1 x_{*2} - \lambda_3 + \lambda_4 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \end{bmatrix}$$

The KKT optimality conditions of the problem are:

$$\left.\begin{array}{c} x_{*1} = \frac{4+\lambda_2-\lambda_4}{2+2\lambda_1} \\ x_{*2} = \frac{4+\lambda_3-\lambda_4}{2+2\lambda_1} \end{array}\right\} \tag{1}$$

$$\lambda_i \geq 0, i = 1, 2, 3, 4 \tag{2}$$

$$x_{*1}^2 + x_{*2}^2 \leq 1 \tag{3}$$

$$x_{*1} \geq 0 \tag{4}$$

$$x_{*2} \geq 0 \tag{5}$$

$$x_{*1} + x_{*2} \leq 1 \tag{6}$$

$$\lambda_1(x_{*1}^2 + x_{*2}^2 - 1) = 0 \tag{7}$$

$$\lambda_2 x_{*1} = 0 \tag{8}$$

$$\lambda_3 x_{*2} = 0 \tag{9}$$

$$\lambda_4(x_{*1} + x_{*2} - 1) = 0 \tag{10}$$

Let's first assume that $x_{*1} = x_{*2} = 0$. Then (3), (4), (5), (6), (8) and (9) hold. In order for (7) and (10) to hold it must be the case that $\lambda_1 = 0$ and $\lambda_4 = 0$. The expression $x_{*1} = x_{*2} = \frac{4+\lambda_2}{2} = \frac{4+\lambda_3}{2} = 0$ follows, leading us to the result $\lambda_2 = \lambda_3 = -4$ which is clearly wrong.

Hence $x_{*1} \neq 0$, $x_{*2} \neq 0$, $\lambda_2 = \lambda_3 = 0$. Using this information to condition (1) we get that

$$x_{*1} = x_{*2} = \frac{4 - \lambda_4}{2 + 2\lambda_1} > 0 \tag{11}$$

$$\lambda_4 \geq 0, \lambda_4 < 4 \tag{12}$$

in order for the conditions conditions (4) and (5) to hold. Know we derive a rule for (6) to hold.

$$x_{*1} + x_{*2} - 1 \leq 0 \Leftrightarrow \frac{8 - 2\lambda_4 - 2 - 2\lambda_1}{2 + 2\lambda_1} \leq 0$$

The denominator is always positive, so

$$8 - 2\lambda_4 - 2 - 2\lambda_1 \leq 0 \Leftrightarrow \lambda_1 \geq 3 - \lambda_4 \tag{13}$$

$$\lambda_4 \in [3, 4) \tag{14}$$

Forming the condition (3) we get

$$0 \geq 2\frac{16 - 8\lambda_4 + \lambda_4^2}{4 + 8\lambda_1 + 4\lambda_1^2} - 1 = \frac{2\lambda_4^2 - 16\lambda_4 + 32 - 4\lambda_1^2 - 8\lambda_1 - 4}{4 + 8\lambda_1 + 4\lambda_1^2}$$

For condition (7) to hold either λ_1 or the nominator of the above relation must be zero. Checking the nominator we get that

$$2\lambda_4^2 - 16\lambda_4 + 32 = 4\lambda_1^2 + 8\lambda_1 + 4$$

$$\Leftrightarrow 2(\lambda_4 - 4)^2 = 4(\lambda_1 + 1)^2$$

$$\Leftrightarrow (\lambda_4 - 4)^2 = 2(\lambda_1 + 1)^2.$$

Given the condition (13), we plug into the above relation the lowest possible value of λ_1 and get that $(\lambda_4 - 4)^2 = 2(\lambda_4 - 4)^2$ which is not possible because of (14). Hence the nominator can never be equal to zero, meaning that $\lambda_1 = 0$. Concluding, we have that

$$x_{*1} = x_{*2} = \frac{4 - \lambda_4}{2} > 0$$

$$\lambda_1 = \lambda_2 = \lambda_3 = 0$$

$$\lambda_4 \in [3, 4)$$

This means that constraint (6) must be active in order for the condition (10) to hold. The condition (6) is active if (13) is active, meaning that $\lambda_4 = 3$, giving us our optimal solution point $x = (\frac{1}{2}, \frac{1}{2})$, as all the KKT conditions hold.

(b) Let $f_0(x) = (x_1+2)^2 + (x_2+2)^2$. Then the convex problem can be rewritten as

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & f_0(x) = (x_1+2)^2 + (x_2+2)^2 \\ \text{subject to} \quad & x_1^2 + x_2^2 - 1 \leq 0 \\ & -x_1 \leq 0 \\ & -x_2 \leq 0 \\ & x_1 + x_2 \leq 0. \end{aligned}$$

The gradients of f_1, f_2, f_3 and f_4 are the same as before.

$$\nabla f_0(x) = \begin{bmatrix} 2(x_{*1}+2) \\ 2(x_{*2}+2) \end{bmatrix}$$

Using the above and the first KKT condition for optimality, we get that

$$\left.\begin{aligned} x_{*1} &= \tfrac{-4+\lambda_2-\lambda_4}{2+2\lambda_1} \\ x_{*2} &= \tfrac{-4+\lambda_3-\lambda_4}{2+2\lambda_1} \end{aligned}\right\}. \tag{1}$$

All the other KKT conditions, from (2) to (10), are the same as before. Lets assume, again, that $x_{*1} = x_{*2} = 0$. Then the conditions (3), (4), (5), (6), (8) and (9) hold. In order for (7) and (10) to hold it must be true that $\lambda_1 = \lambda_4 = 0$. Then we get $x_{*1} = x_{*2} = \frac{-4+\lambda_2}{2} = \frac{-4+\lambda_3}{2} = 0$ which means that $\lambda_2 = \lambda_3 = 4$. So all the KKT conditions hold, meaning that the point $x = (0,0)$ is the optimal solution of this convex problem.

3. Consider the problem

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & f(x) = -\sum_{i=1}^{n} log(x_i) \\ \text{subject to} \quad & \boldsymbol{Ax} = \boldsymbol{b} \end{aligned}$$

where $A \in \mathbb{R}_+^{p\times n}$, $rank(A) = p$, $b \in \mathbb{R}_+^p$. This is a convex optimization problem with affine constraints. The problem is solved in four different ways, in the matlab code implementation below. The first solution is given by the cvx library. The second solution is reached via Newton, starting from a feasible point computed from cvx. The algorithm can be seen below.

$\boldsymbol{x} \in \boldsymbol{dom}$ f, $\boldsymbol{Ax} = \boldsymbol{b}$, tolerance ϵ

while *TRUE* **do**

1. compute Newton step and decrement : $\Delta_{\boldsymbol{x}_{N_t}}, \lambda(\boldsymbol{x})$.
2. quit if $\frac{\lambda^2(\boldsymbol{x})}{2} \leq \epsilon$.
3. Perform backtracking line search and choose t.
4. $\boldsymbol{x} := \boldsymbol{x} + t\Delta_{\boldsymbol{x}_{N_t}}$.

end

Algorithm 1: Newtons' algorithm starting from a feasible point.

with

$$w = -\Big(A\Big(\nabla^2 f(x)\Big)^{-1}A^T\Big)^{-1}A\Big(\nabla^2 f(x)\Big)^{-1}\nabla f(x)$$

$$\Delta_{x_{N_t}} = -\Big(\nabla^2 f(x)\Big)^{-1}\Big(\nabla f(x) + A^T w\Big)$$

$$\lambda^2(x) = -\nabla f(x)^T \Delta_{x_{N_t}}$$

$$\nabla f(x) = \begin{bmatrix} -x_1^{-1} & ..... & -x_n^{-1} \end{bmatrix}^T$$

$$\nabla^2 f(x) = diag(\begin{bmatrix} x_1^{-2} & ..... & x_n^{-2} \end{bmatrix}^T).$$

The third solution is reached via the primal-dual algorithm, starting from point $x_0 = \mathbf{1}$. The algorithm can be seen below.

$\boldsymbol{x} \in \boldsymbol{dom}$ f, $\boldsymbol{v} \in \mathbb{R}^p$, tolerance ϵ

repeat

1. compute primal and dual Newton steps : $\Delta_{\boldsymbol{x}_{pd}}, \Delta_{\boldsymbol{v}_{pd}}$.
2. Backtracking line search

   $t := 1$

   while $||r(\boldsymbol{x} + \Delta_{\boldsymbol{x}_{pd}}, \boldsymbol{v} + \Delta_{\boldsymbol{v}_{pd}})||_2 > (1 - at)||r(\boldsymbol{x}, \boldsymbol{v})||_2$ **do**

   $t := \beta t$

   end
3. $\boldsymbol{x} := \boldsymbol{x} + \Delta_{\boldsymbol{x}_{pd}}$, $\boldsymbol{v} := \boldsymbol{v} + \Delta_{\boldsymbol{v}_{pd}}$.

until $||r(\boldsymbol{x}, \boldsymbol{v})||_2 \leq \epsilon$;

Algorithm 2: Newtons' algorithm starting from a non-feasible point.

where

$$v_+ = -\Big(A\Big(\nabla^2 f(x)\Big)^{-1}A^T\Big)^{-1}\Big(A\Big(\nabla^2 f(x)\Big)^{-1}\nabla f(x) - Ax + b\Big)$$

$$\Delta_{v_{pd}} = v_+ - v$$

$$\Delta_{x_{pd}} = -\Big(\nabla^2 f(x)\Big)^{-1}\Big(\nabla f(x) + A^T v_+\Big)$$

and $\nabla f(x)$, $\nabla^2 f(x)$ are the same as before. Finally, the forth solution is reached by solving the dual problem via cvx. The uncostrained problem solved by the cvx library is

$$\underset{v\in\mathbb{R}^p}{\text{minimize}} \quad f(v) = -\sum_{i=1}^{n} log\Big((A^T v)_i\Big) - n + b^T v.$$

We get the optimal point x_* by substituting the optimal Lagrange multipliers v_{*i} to the relation $x_{*i} = \frac{1}{(A^T v)_i}$. All four solutions converge approximately to the same point.

Matlab implementation

Main.m

```
breaklines
% Vissarion Konidaris
% Convex Optimization Course
% 6/5/2018

clear ; close all; clc
format long;

disp('Minimization problem');
disp('-sum(log(x))');
disp('subject to Ax=b');

in=0;
while(in~=1 && in~=2 && in~=3)
  disp([newline 'Give the dimensions of the problem.']);
  disp('Indicative value pairs are (n,m) = (2,50), ');
```

```
  disp('(200,50),(800,300) where m the number of the ');
  disp('logarithms and n the dimension of input x.');
  disp('Type 1 for (2,50), 2 for (5,200), 3 for (800,300).');
  in=input('n : ');
  if(in==1)
    n=20;
    p=2;
  elseif(in==2)
    n=200;
    p=50;
  else
    n=800;
    p=300;
  end
  disp('');
end

% The acceptable error of the optimizer.
e = -1.;
while(e<0.)
  disp([newline 'Give the acceptable error of the optimizer.']);
  disp('Acceptable number of error are ');
  disp('the all the positive real numbers.');
  e=input('e : ');
  disp('');
end

aplha=0.5;
beta=1.;
while(aplha>=0.5 || aplha<=0.)
  disp([newline 'Give a value for the parameter aplha.']);
  disp('Acceptable values are in the open interval (0,0.5).');
  aplha=input('aplha : ');
  disp('');
end
while(beta>=1. || beta<=0.)
  disp('Give a value for the parameter beta.');
  disp('Acceptable values are in the open interval (0,1).');
  beta=input('beta : ');
  disp('');
```

```
end

A=rand(p,n);
x=rand(n,1);
b=A*x;

%%% Cvx solution %%%

cvx_begin
    variable cv_x(n);
    minimize (-sum(log(cv_x)));
    subject to
        A*cv_x-b==0;
        -cv_x<0;
cvx_end
cv_min_fun_val=-sum(log(cv_x));
disp(cv_min_fun_val);


%%% Newtons' method starting from a feasible point %%%

% Gradient Descent using back-tracking line search.
disp([newline 'Newtons method starting from a feasible point.']);

% Finding a feasible starting point x0 via cvx
cvx_begin
    variable x0(n);
    minimize(0);
    subject to
        A*x0-b==0;
        -x0<0;
cvx_end

x=x0;
iter=0;
while(1)
    grad=-x.^-1;
    hessian=diag(x.^-2);
    hessian_inv=inv(hessian);
    w=-inv(A*hessian_inv*A')*A*hessian_inv*grad;
```

```
    delta=-hessian_inv*(grad+A'*w);
    lambda_squared=-grad'*delta;

    if(lambda_squared/2<=e)
        break;
    end

    t=1;
    while(1)
        feasible=1;
        point = x+t*delta;
        for i=1:size(point,1)
            if(point(i,1)<=0)
                feasible=0;
                break;
            end
        end
        if(feasible==0)
            disp(['Not in domain at loop ',num2str(iter+1)]);
            t=beta*t;
            continue;
        end
        break;
    end
    f=-sum(log(x));
    while(-sum(log(x+t*delta))...
        > f-aplha*t*lambda_squared)
        t=beta*t;
    end

    iter=iter+1;
    x=x+t*delta;
end
disp(['Actual iterations : ',num2str(iter)]);
disp(['Error : ',num2str(norm(cv_x-x))]);
disp(['Optimal value (cvx_optval): ',...
      num2str(cv_min_fun_val)]);
disp(['Optimal value (Newtons): ',...
      num2str(-sum(log(x)))]);
disp('')
```

```
%%% Newtons' method starting from a non feasible point %%%
disp([newline 'Primal-Dual algorithm starting from  a non feasible point.']);

x=ones(n,1);
v=rand(p,1);
iter=0;
stopping_norm=1.;
while(stopping_norm>e)
    grad=-x.^-1;
    hessian=diag(x.^-2);
    hessian_inv=inv(hessian);
    v_plus=-inv(A*hessian_inv*A')*(A*hessian_inv*grad-A*x+b);
    v_d=v_plus-v;
    x_d=-hessian_inv*(grad+A'*v_plus);

    t=1;
    while(1)
        feasible=1;
        point = x+t*x_d;
        for i=1:size(point,1)
            if(point(i,1)<=0)
                feasible=0;
                break;
            end
        end
        if(feasible==0)
            disp(['Not in domain at loop ',num2str(iter+1)]);
            t=beta*t;
            continue;
        end
        break;
    end
    f=sqrt(norm(grad+A'*v,2)^2+norm(A*x-b,2)^2);
    while(sqrt(...
          norm(-(x+t*x_d).^(-1)+A'*(v+t*v_d),2)^2+...
          norm(A*(x+t*x_d)-b,2)^2)...
        > (1-aplha*t)*f)
        t=beta*t;
    end
```

```
    iter=iter+1;
    x=x+t*x_d;
    v=v+t*v_d;
    stopping_norm=sqrt(norm(-x.^-1+A'*v,2)^2+norm(A*x-b,2)^2);
end
disp(['Actual iterations : ',num2str(iter)]);
disp(['Error : ',num2str(norm(cv_x-x))]);
disp(['Optimal value (cvx_optval): ',...
      num2str(cv_min_fun_val)]);
disp(['Optimal value (Newtons): ',...
      num2str(-sum(log(x)))]);
disp('');

%%% Cvx dual solution %%%
disp([newline 'Solving the Dual problem via cvx.']);
cvx_begin
    variable lag_mul(p);
    minimize (-sum(log(A'*lag_mul))-n+b'*lag_mul);
cvx_end
cvx_dual_sol=(A'*lag_mul).^-1;
cv_min_fun_val_dual=-sum(log(cvx_dual_sol));
disp(cv_min_fun_val_dual);
```