<u>ESISTENZA DI OTTIMI GLOBALI E CONVESSITA'</u>

1) Esistenza ottimo globale

<u>Consideriamo problemi di programmazione lineare:</u>

- **<u>Weirestrass</u>**: se la regione di fattibilità è chiusa e limitata e la funzione obiettivo è continua, allora esiste un ottimo globale.
- Se la funzione obiettivo f è continua e **coerciva** ($\lim_{||x||\to\infty} f(x) = +\infty$) e la regione di fattibilità Ω è chiusa, allora esiste un ottimo globale
- Se **f è strongly convex e Ω è chiusa** allora esiste un ottimo globale
- Se **f è strongly convex e Ω è chiusa e convessa** allora esiste un UNICO ottimo globale

<u>Consideriamo problemi di programmazione quadratica:</u>

- **Teorema di Eaves**: (P) ha ottimi globali se e solo se sono soddisfatte le seguenti condizioni:
    - $d^TQd \geq 0$, per ogni d ∈ rec(Ω)
    - $d^T(Qx+c) \geq 0$, per ogni x ∈ Ω ed ogni d ∈ rec(Ω) tale che $d^TQd = 0$

Dove rec(Ω) = { d: Ad ≤ 0}

Casi speciali del teorema di Eaves sono:
- Se Q = 0 allora (P) ha un ottimo globale se e solo se $d^Tc \geq 0$, per ogni d ∈ rec(Ω)
- Se Q è definita positiva allora il teorema è soddisfatto
- Se Ω è limitata allora il teorema è soddisfatto

2) Convessità

<u>Nel caso in cui la funzione è quadratica</u>, dobbiamo vedere la sua hessiana Q:
- **Semidefinita positiva** -> problema convesso
- **Definita positiva** -> problema strongly convex

<u>Se la funzione è lineare,</u> **è convesso ma non strongly convex.**

SVM

1) Soft margin

Il problema può essere scritto:

$$\begin{cases} \min\ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell} \xi_i \\ 1 - y^i(w^\top x^i + b) \leq \xi_i & \forall\ i = 1, \dots, \ell \\ \xi_i \geq 0 & \forall\ i = 1, \dots, \ell \end{cases}$$

Il duale invece nel seguente modo:

$$\begin{cases} \max\ -\frac{1}{2}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell} y^i y^j (x^i)^\top x^j \lambda_i \lambda_j + \sum_{i=1}^{\ell} \lambda_i \\ \sum_{i=1}^{\ell} \lambda_i y^i = 0 \\ 0 \leq \lambda_i \leq C \qquad i = 1, \dots, \ell \end{cases}$$

If λ^* is dual optimum, then

$$w^* = \sum_{i=1}^{\ell} \lambda_i^* y^i x^i.$$

Find b^* choosing i s.t. $0 < \lambda_i^* < C$ and using the complementarity conditions:

$$\begin{cases} \lambda_i^* \left[1 - y^i((w^*)^\top x^i + b^*) - \xi_i^*\right] = 0 \\ (C - \lambda_i^*)\xi_i^* = 0 \end{cases}$$

Thus $b^* = \dfrac{1}{y^i} - (w^*)^\top x^i$

Per risolvere il problema su MATLAB dobbiamo definire inoltre:

$$Q = y^i y^j x^i (x^j)'$$
$$y = [1, \dots, 1, -1, \dots, -1]$$

Il numero di 1 e -1 sono quanti la dimensione del dataset trovata precedentemente (nA e nB).
Definiti i dataset A e B il codice è il seguente:

```
nA = size(A,1);
nB = size(B,1);

% training points
T = [A ; B];

%% Linear SVM - dual model (soft margin)

% define the problem
C = 10 ;
y = [ones(nA,1) ; -ones(nB,1)]; % labels
l = length(y);
Q = zeros(l,l);
for i = 1 : l
    for j = 1 : l
        Q(i,j) = y(i)*y(j)*(T(i,:))*T(j,:)' ;
    end
end

% solve the problem
options = optimset('Largescale','off','display','off');
la = quadprog(Q,-ones(l,1),[],[],y',0,zeros(l,1),C*ones(l,1),[],options);

% compute vector w
wD = zeros(2,1);
for i = 1 : l
   wD = wD + la(i)*y(i)*T(i,:)';
end
wD

% compute scalar b
indpos = find(la > 1e-3);
ind = find(la(indpos) < C - 1e-3);
i = indpos(ind(1));
bD = 1/y(i) - wD'*T(i,:)'

%% plot the solution
xx = 0:0.1:10;
uuD = (-wD(1)/wD(2)).*xx - bD/wD(2);
vvD = (-wD(1)/wD(2)).*xx + (1-bD)/wD(2);
vvvD = (-wD(1)/wD(2)).*xx + (-1-bD)/wD(2);
plot(A(:,1),A(:,2),'bo',B(:,1),B(:,2),'ro',...
    xx,uuD,'k-',xx,vvD,'b-',xx,vvvD,'r-','Linewidth',1.5)
axis([0 10 0 10])
title('Optimal separating hyperplane (soft margin)')
```

2) Kernel function

Il problema ora è il seguente:

Primal problem:

$$\begin{cases} \min \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell} \xi_i \\ 1 - y^i(w^\mathsf{T}\phi(x^i) + b) \le \xi_i & \forall\ i = 1,\dots,\ell \\ \xi_i \ge 0 & \forall\ i = 1,\dots,\ell \end{cases}$$

w is a vector in a high dimensional space (maybe infinite variables)

Dual problem is convex:

$$\begin{cases} \max \ -\frac{1}{2}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell} y^i y^j k(x^i, x^j)\lambda_i\lambda_j + \sum_{i=1}^{\ell}\lambda_i \\ \sum_{i=1}^{\ell}\lambda_i y^i = 0 \\ 0 \le \lambda_i \le C \qquad i = 1,\dots,\ell \end{cases}$$

I risultati sono:

- choose a kernel k
- solve the dual $\to \lambda^*$
- choose i s.t. $0 < \lambda_i^* < C$ and find b^*:

$$b^* = \frac{1}{y^i} - \sum_{j=1}^{\ell} \lambda_j^* y^j k(x^i, x^j)$$

- Decision function

$$f(x) = \text{sign}\left(\sum_{i=1}^{\ell} \lambda_i^* y^i k(x^i, x) + b^*\right)$$

Le funzioni kernel possono essere:

- $k(x,y) = x^\mathsf{T} y$
- $k(x,y) = (x^\mathsf{T} y + 1)^p$, with $p \ge 1$ (polynomial)
- $k(x,y) = e^{-\gamma\|x-y\|^2}$ (Gaussian)
- $k(x,y) = \tanh(\beta x^\mathsf{T} y + \gamma)$, with suitable β and γ

Definiti i data set A e B avremo il seguente codice:

```
nA = size(A,1);
nB = size(B,1);

% training points
T = [A ; B];
y = [ones(nA,1) ; -ones(nB,1)]; % labels
l = length(y);
%% Nonlinear SVM

% parameter
C = 1 ;

% Gaussian kernel
gamma = 1 ;
K = zeros(l,l);
for i = 1 : l
    for j = 1 : l
        K(i,j) = exp(-gamma*norm(T(i,:)-T(j,:))^2);
    end
end

% define the problem
Q = zeros(l,l);
for i = 1 : l
    for j = 1 : l
        Q(i,j) = y(i)*y(j)*K(i,j) ;
    end
end

% solve the problem
[la,ov] = quadprog(Q,-ones(l,1),[],[],y',0,zeros(l,1),C*ones(l,1));

% compute b
ind = find(la > 1e-3 & la < C - 1e-6);
i = ind(1);
b = 1/y(i) ;
for j = 1 : l
    b = b - la(j)*y(j)*K(i,j);
end
b

%% plot the surface
AA=[];
BB=[];
for xx = -2 : 0.05 : 2
    for yy = -2 : 0.05 : 2
        s = 0;
        for i = 1 : l
            s = s + la(i)*y(i)*exp(-gamma*norm(T(i,:)-[xx yy])^2);
        end
        s = s + b;
        if s > 0
            AA = [AA ; xx yy];
        else
            BB = [BB ; xx yy];
        end
    end
end
plot(A(:,1),A(:,2),'bo',B(:,1),B(:,2),'ro','Linewidth',5)
hold on
plot(AA(:,1),AA(:,2),'b.',BB(:,1),BB(:,2),'r.','Linewidth',0.01);
title(['Separating surface with C = ',num2str(C),' and \gamma =
',num2str(gamma)])
```

REGRESSION

Considerando il seguente polinomio:

$$p(x) = z_1 + z_2\,x + z_3\,x^2 + \cdots + z_n\,x^{n-1}$$

Avremo che $r_i = p(x) - y_i$ e dovremmo minimizzare r_i.

1)

Euclidean norm $\|\cdot\|_2$ (least squares approximation) $\rightarrow$ quadratic programming problem:

$$\begin{cases} \min\ \frac{1}{2}\|Az - y\|_2^2 = \frac{1}{2}(Az-y)^\mathsf{T}(Az-y) = \frac{1}{2}z^\mathsf{T}A^\mathsf{T}Az - z^\mathsf{T}A^\mathsf{T}y + \frac{1}{2}y^\mathsf{T}y \\ z \in \mathbb{R}^n \end{cases}$$

It can be proved that $\text{rank}(A) = n$, thus $A^\mathsf{T}A$ is positive definite.

Hence, the unique optimal solution is the stationary point of the objective function, i.e., the solution of the system of linear equations:

$$A^\mathsf{T}Az = A^\mathsf{T}y$$

2)

Norm $\|\cdot\|_1 \rightarrow$ linear programming problem:

$$\begin{cases} \min\ \|Az - y\|_1 = \sum_{i=1}^{\ell} |A_i x - y_i| \\ z \in \mathbb{R}^n \end{cases}$$

is equivalent to

$$\begin{cases} \min\ \sum_{i=1}^{\ell} u_i \\ u_i = |A_i z - y_i| \\ \quad = \max\{A_i z - y_i, y_i - A_i z\} \end{cases} \rightarrow \begin{cases} \min\ \sum_{i=1}^{\ell} u_i \\ u_i \geq A_i z - y_i & \forall\ i = 1,\dots,\ell \\ u_i \geq y_i - A_i z & \forall\ i = 1,\dots,\ell \end{cases}$$

3)

Norm $\|\cdot\|_\infty \rightarrow$ linear programming problem:

$$\begin{cases} \min\ \|Az - y\|_\infty = \max_{i=1,\dots,\ell} |A_i z - y_i| \\ z \in \mathbb{R}^n \end{cases}$$

is equivalent to

$$\begin{cases} \min\ u \\ u \geq A_i z - y_i & \forall\ i = 1,\dots,\ell \\ u \geq y_i - A_i z & \forall\ i = 1,\dots,\ell \end{cases}$$

Considerando il dataset data:

```
x = data(:,1) ;
y = data(:,2) ;
l = length(x) ;
n = 4 ;

% Vandermonde matrix (grado 3)
A = [ ones(l,1) x x.^2 x.^3 ];

%% 2-norm problem

z2 = (A'*A)\(A'*y)
p2 = A*z2;

%% 1-norm problem

% define the problem
c = [ zeros(n,1); ones(l,1) ];
D = [ A -eye(l); -A -eye(l) ];
d = [ y; -y ];

% solve the problem
sol1 = linprog(c,D,d) ;
z1 = sol1(1:n)
p1 = A*z1;

%% inf-norm problem

% define the problem
c = [ zeros(n,1); 1 ];
D = [ A -ones(l,1); -A -ones(l,1) ];

% solve the problem
solinf = linprog(c,D,d) ;
zinf = solinf(1:n)
pinf = A*zinf;

%% plot the solutions

plot(x,y,'b.',x,p2,'r-',x,p1,'k-',x,pinf,'g-')
legend('Data','2-norm','1-norm','inf-norm',...
    'Location','NorthWest');
```

4) Linear SVM con variabili slack

$$
\begin{cases}
\min \ \dfrac{1}{2}\|w\|^2 + C\displaystyle\sum_{i=1}^{\ell}(\xi_i^+ + \xi_i^-) \\
y_i \leq w^T x_i + b + \varepsilon + \xi_i^+ & \forall\ i = 1,\dots,\ell \\
y_i \geq w^T x_i + b - \varepsilon - \xi_i^- & \forall\ i = 1,\dots,\ell
\end{cases}
$$

Considerando come dataset data:

```
x = data(:,1) ;
y = data(:,2) ;
l = length(x) ; % number of points

%% linear regression - primal problem with slack variables

% parameters
epsilon = 0.2 ;
C = 10 ;

% define the problem
Q = [ 1               zeros(1,2*l+1)
      zeros(2*l+1,1) zeros(2*l+1) ];

c = [ 0 ; 0 ; C*ones(2*l,1)];

D = [-x -ones(l,1) -eye(l)    zeros(l)
      x  ones(l,1)  zeros(l) -eye(l)];

d = epsilon*ones(2*l,1) + [-y;y];

% solve the problem
sol = quadprog(Q,c,D,d,[],[],[-inf;-inf;zeros(2*l,1)],[]);

% compute w
w = sol(1);

% compute b
b = sol(2);

% compute clask variables xi+ and xi-
xip = sol(3:2+l);
xim = sol(3+l:2+2*l);

% find regression and epsilon-tube
z = w.*x + b ;
zp = w.*x + b + epsilon ;
zm = w.*x + b - epsilon ;

%% plot the solution

plot(x,y,'b.',x,z,'k-',x,zp,'r-',x,zm,'r-');
legend('Data','regression',...
    '\epsilon-tube','Location','NorthWest')
```

5) Nonlinear SVM

Primal problem:

$$\begin{cases} \min \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell}(\xi_i^+ + \xi_i^-) \\ y_i \leq w^T\phi(x_i) + b + \varepsilon + \xi_i^+ & \forall\ i = 1,\dots,\ell \\ y_i \geq w^T\phi(x_i) + b - \varepsilon - \xi_i^- & \forall\ i = 1,\dots,\ell \end{cases}$$

w is a vector in a high dimensional space (maybe infinite variables)

Dual problem:

$$\begin{cases} \max\limits_{(\lambda^+,\lambda^-)} \ -\frac{1}{2}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}(\lambda_i^+ - \lambda_i^-)(\lambda_j^+ - \lambda_j^-)k(x_i, x_j) \\ \qquad -\varepsilon\sum_{i=1}^{\ell}(\lambda_i^+ + \lambda_i^-) + \sum_{i=1}^{\ell} y_i(\lambda_i^+ - \lambda_i^-) \\ \sum_{i=1}^{\ell}(\lambda_i^+ - \lambda_i^-) = 0 \\ \lambda_i^+, \lambda_i^- \in [0, C] \end{cases}$$

number of variables $= 2\ell$

Therefore:

- choose a kernel k
- solve the dual $\to (\lambda^+, \lambda^-)$
- find b:

$$b = y_i - \varepsilon - \sum_{j=1}^{\ell}(\lambda_j^+ - \lambda_j^-)k(x_i, x_j), \qquad \text{for some } i \text{ s.t. } 0 < \lambda_i^+ < C$$

or

$$b = y_i + \varepsilon - \sum_{j=1}^{\ell}(\lambda_j^+ - \lambda_j^-)k(x_i, x_j), \qquad \text{for some } i \text{ s.t. } 0 < \lambda_i^- < C$$

- Recession function

$$f(x) = \sum_{i=1}^{\ell}(\lambda_i^+ - \lambda_i^-)k(x^i, x) + b$$

Consider dataset data:

```
x = data(:,1) ;
y = data(:,2) ;
l = length(x) ; % number of points

%% nonlinear regression - dual problem
epsilon = 10 ;
C = 10;
kernel_type = 2;

% define the problem
Q = zeros(2*l);
for i = 1 : l
    for j = 1 : l
        Q(i,j) = kernel(x(i),x(j));
    end
end
for i = l+1 : 2*l
    for j = l+1 : 2*l
        Q(i,j) = kernel(x(i-l),x(j-l));
```

```
    end
end
for i = 1 : l
    for j = l+1 : 2*l
        Q(i,j) = -kernel(x(i),x(j-l));
    end
end
for i = l+1 : 2*l
    for j = 1 : l
        Q(i,j) = -kernel(x(i-l),x(j));
    end
end

c = epsilon*ones(2*l,1) + [-y;y];

% solve the problem
sol = quadprog(Q,c,[],[],...
    [ones(1,l) -ones(1,l)],0,...
    zeros(2*l,1),C*ones(2*l,1));
lap = sol(1:l);
lam = sol(l+1:2*l);

% compute b
ind = find(lap > 1e-3 & lap < C-1e-3);
if ~isempty(ind)
    i = ind(1);
    b = y(i) - epsilon;
    for j = 1 : l
        b = b - (lap(j)-lam(j))*kernel(x(i),x(j));
    end
else
    ind = find(lam > 1e-3 & lam < C-1e-3);
    i = ind(1);
    b = y(i) + epsilon ;
    for j = 1 : l
        b = b - (lap(j)-lam(j))*kernel(x(i),x(j));
    end
end

% find regression and epsilon-tube
z = zeros(l,1);
for i = 1 : l
    z(i) = b ;
    for j = 1 : l
        z(i) = z(i) + (lap(j)-lam(j))*kernel(x(i),x(j));
    end
end
zp = z + epsilon ;
zm = z - epsilon ;

%% plot the solution

% find support vectors
sv = [find(lap > 1e-3);find(lam > 1e-3)];
sv = sort(sv);

plot(x,y,'b.',x(sv),y(sv),...
    'ro',x,z,'k-',x,zp,'r-',x,zm,'r-');
legend('Data','Support vectors',...
    'regression','\epsilon-tube',...
    'Location','NorthWest')
```

KERNEL

```
function v = kernel(x,y)

global kernel_type

if kernel_type == 1 % gaussian
    gamma = 1 ;
    v = exp(-gamma*norm(x-y)^2);
end

if kernel_type == 2 % polynomial
    p = 3 ;
    v = (x'*y + 1)^p;
end

if kernel_type == 3 % hyperbolic tangent
    theta = 0;
    k = 1;
    v = tanh(theta + k*x'*y) ;
end

end
```

CLUSTERING

1) MODELLO CON ||-||₂

$$\begin{cases} \min \sum_{i=1}^{\ell} \min_{j=1,\dots,k} \|p_i - x_j\|_2^2 \\ x_j \in \mathbb{R}^n \qquad \forall\, j = 1,\dots,k \end{cases}$$

<u>Se k = 1:</u>

$$\begin{cases} \min \sum_{i=1}^{\ell} \|p_i - x\|_2^2 = \sum_{i=1}^{\ell} (x - p_i)^T (x - p_i) \\ x \in \mathbb{R}^n \end{cases}$$

The global optimum is the stationary point:

$$2\ell x - 2\sum_{i=1}^{\ell} p_i = 0 \quad \Longleftrightarrow \quad x = \frac{\sum_{i=1}^{\ell} p_i}{\ell} \quad \text{(mean or baricenter)}$$

<u>Se k>1:</u>

Problem (2) is equivalent to the following nonconvex smooth problem:

$$\begin{cases} \min_{x,\alpha} \sum_{i=1}^{\ell} \sum_{j=1}^{k} \alpha_{ij} \|p_i - x_j\|_2^2 \\ \sum_{j=1}^{k} \alpha_{ij} = 1 \quad \forall\, i = 1,\dots,\ell \\ \alpha_{ij} \geq 0 \qquad \forall\, i = 1,\dots,\ell,\ j = 1,\dots,k \\ x_j \in \mathbb{R}^n \qquad \forall\, j = 1,\dots,k. \end{cases}$$

Algoritmo k-means:

0. (Inizialization) Set $t = 0$, choose centroids $x_1^0, \dots, x_k^0 \in \mathbb{R}^n$ and assign patterns to clusters: for any $i = 1,\dots,\ell$

$$\alpha_{ij}^0 = \begin{cases} 1 & \text{if } j \text{ is the first index s.t. } \|p_i - x_j^0\|_2 = \min_{h=1,\dots,k} \|p_i - x_h^0\|_2 \\ 0 & \text{otherwise.} \end{cases}$$

1. (Update centroids) For each $j = 1,\dots,k$ compute the mean

$$x_j^{t+1} = \left(\sum_{i=1}^{\ell} \alpha_{ij}^t p_i \right) / \left(\sum_{i=1}^{\ell} \alpha_{ij}^t \right).$$

2. (Update clusters) For any $i = 1,\dots,\ell$ compute

$$\alpha_{ij}^{t+1} = \begin{cases} 1 & \text{if } j \text{ is the first index s.t. } \|p_i - x_j^{t+1}\|_2 = \min_{h=1,\dots,k} \|p_i - x_h^{t+1}\|_2 \\ 0 & \text{otherwise.} \end{cases}$$

3. (Stopping criterion) If $f(x^{t+1}, \alpha^{t+1}) = f(x^t, \alpha^t)$ then STOP
   else $t = t + 1$ go to Step 1.

Considerando il dataset data:

```
l = size(data,1); % number of patterns

% plot patterns
plot(data(:,1),data(:,2),'ko');
axis([0 10 0 10])
title('k-means algorithm');
hold on

k = 3; % number of clusters

% initialize centroids
x = [5 7; 6 3 ; 4 3];  % part a)

% plot centroids
plot(x(1,1),x(1,2),'b^',...
    x(2,1),x(2,2),'r^',...
    x(3,1),x(3,2),'g^');

pause

% initialize clusters
cluster = zeros(l,1);
for i = 1 : l
    d = inf;
    for j = 1 : k
        if norm(data(i,:)-x(j,:)) < d
            d = norm(data(i,:)-x(j,:));
            cluster(i) = j;
        end
    end
end

% plot cluster
c1 = data(cluster==1,:);
c2 = data(cluster==2,:);
c3 = data(cluster==3,:);
plot(c1(:,1),c1(:,2),'bo',c2(:,1),c2(:,2),'ro',...
    c3(:,1),c3(:,2),'go');

% compute the objective function value
v = 0;
for i = 1 : l
    v = v + norm(data(i,:)-x(cluster(i),:))^2 ;
end
title(['k-means algoritm: objective function = ',num2str(v)]);

pause

while true

    % delete old centroids
    plot(x(1,1),x(1,2),'w^',...
        x(2,1),x(2,2),'w^',...
        x(3,1),x(3,2),'w^');

    % update centroids
    for j = 1 : k
        ind = find(cluster == j);
        if ~isempty(ind)
            x(j,:) = mean(data(ind,:));
        end
    end
```

```
    % plot new centroids
    plot(x(1,1),x(1,2),'b^',...
        x(2,1),x(2,2),'r^',...
        x(3,1),x(3,2),'g^');

    pause

    % update clusters
    for i = 1 : l
        d = inf;
        for j = 1 : k
            if norm(data(i,:)-x(j,:)) < d
                d = norm(data(i,:)-x(j,:));
                cluster(i) = j;
            end
        end
    end

    % plot cluster
    c1 = data(cluster==1,:);
    c2 = data(cluster==2,:);
    c3 = data(cluster==3,:);
    plot(c1(:,1),c1(:,2),'bo',c2(:,1),c2(:,2),...
        'ro',c3(:,1),c3(:,2),'go');


    % update objective function
    vnew = 0;
    for i = 1 : l
        vnew = vnew + norm(data(i,:)-x(cluster(i),:))^2 ;
    end
    title(['k-means algoritm: objective function = ',num2str(vnew)]);

    pause

    % stopping criterion
    if v - vnew < 1e-5
        break
    else
        v = vnew;
    end

end

v
```

2) Modello con ||-||₁

$$\begin{cases} \min \sum_{i=1}^{\ell} \min_{j=1,\dots,k} \|p_i - x_j\|_1 \\ x_j \in \mathbb{R}^n \qquad \forall\, j = 1,\dots,k \end{cases}$$

<u>Se k=1:</u>

$$\begin{cases} \min \sum_{i=1}^{\ell} \|p_i - x\|_1 = \sum_{i=1}^{\ell}\sum_{h=1}^{n} |x_h - (p_i)_h| = \sum_{h=1}^{n} \underbrace{\sum_{i=1}^{\ell} |x_h - (p_i)_h|}_{f_h(x_h)} \\ x \in \mathbb{R}^n \end{cases}$$

The global optimum is median$(a_1,\dots,a_\ell) = \begin{cases} a_{(\ell+1)/2} & \text{if } \ell \text{ is odd,} \\ \dfrac{a_{\ell/2} + a_{1+\ell/2}}{2} & \text{if } \ell \text{ is even.} \end{cases}$

<u>Se k>1:</u>

$$\begin{cases} \min_{x,\alpha} \sum_{i=1}^{\ell}\sum_{j=1}^{k} \alpha_{ij}\|p_i - x_j\|_1 \\ \sum_{j=1}^{k} \alpha_{ij} = 1 \quad \forall\, i = 1,\dots,\ell \\ \alpha_{ij} \geq 0 \qquad \forall\, i = 1,\dots,\ell,\ j = 1,\dots,k \\ x_j \in \mathbb{R}^n \qquad \forall\, j = 1,\dots,k. \end{cases}$$

Che può essere scritto come:

$$\begin{cases} \min_{x,\alpha,u} \sum_{i=1}^{\ell}\sum_{j=1}^{k}\sum_{h=1}^{n} \alpha_{ij} u_{ijh} \\ u_{ijh} \geq (p_i)_h - (x_j)_h \qquad \forall\, i = 1,\dots,\ell,\ j = 1,\dots,k,\ \ h = 1,\dots,n \\ u_{ijh} \geq (x_j)_h - (p_i)_h \qquad \forall\, i = 1,\dots,\ell,\ j = 1,\dots,k,\ \ h = 1,\dots,n \\ \sum_{j=1}^{k} \alpha_{ij} = 1 \qquad \forall\, i = 1,\dots,\ell \\ \alpha_{ij} \geq 0 \qquad \forall\, i = 1,\dots,\ell,\ j = 1,\dots,k \\ x_j \in \mathbb{R}^n \qquad \forall\, j = 1,\dots,k. \end{cases}$$

Algoritmo k-median:

0. (Inizialization) Set $t = 0$, choose centroids $x_1^0, \dots, x_k^0 \in \mathbb{R}^n$ and assign patterns to clusters: for any $i = 1, \dots, \ell$

$$\alpha_{ij}^0 = \begin{cases} 1 & \text{if } j \text{ is the first index s.t. } \|p_i - x_j^0\|_1 = \min_{h=1,\dots,k} \|p_i - x_h^0\|_1 \\ 0 & \text{otherwise.} \end{cases}$$

1. (Update centroids) For each $j = 1, \dots, k$ compute

$$x_j^{t+1} = \text{median}(p_i : \ \alpha_{ij}^t = 1).$$

2. (Update clusters) For any $i = 1, \dots, \ell$ compute

$$\alpha_{ij}^{t+1} = \begin{cases} 1 & \text{if } j \text{ is the first index s.t. } \|p_i - x_j^{t+1}\|_1 = \min_{h=1,\dots,k} \|p_i - x_h^{t+1}\|_1 \\ 0 & \text{otherwise.} \end{cases}$$

3. (Stopping criterion) If $f(x^{t+1}, \alpha^{t+1}) = f(x^t, \alpha^t)$ then STOP
else $t = t + 1$ go to Step 1.

Considerando il dataset data:

```
l = size(data,1); % number of patterns
% plot patterns
plot(data(:,1),data(:,2),'ko');
axis([0 10 0 10])
title('k-median algoritm');
hold on
pause

k = 3; % number of clusters

% initialize centroids
x = [5 7; 6 3 ; 4 3];  % part a)

% plot centroids
plot(x(1,1),x(1,2),'b^',...
    x(2,1),x(2,2),'r^',...
    x(3,1),x(3,2),'g^');

pause

% initialize clusters
cluster = zeros(l,1);
for i = 1 : l
    d = inf;
    for j = 1 : k
        if norm(data(i,:)-x(j,:),1) < d
            d = norm(data(i,:)-x(j,:),1);
            cluster(i) = j;
        end
    end
end

% plot cluster
c1 = data(cluster==1,:);
c2 = data(cluster==2,:);
c3 = data(cluster==3,:);
plot(c1(:,1),c1(:,2),'bo',c2(:,1),c2(:,2),'ro',c3(:,1),c3(:,2),'go');

% compute the objective function value
v = 0;
```

```
for i = 1 : l
    v = v + norm(data(i,:)-x(cluster(i),:),1);
end
title(['k-median algoritm: objective function = ',num2str(v)]);
pause

while true

    % delete old centroids
    plot(x(1,1),x(1,2),'w^',...
        x(2,1),x(2,2),'w^',...
        x(3,1),x(3,2),'w^');

    % update centroids
    for j = 1 : k
        ind = find(cluster == j);
        if ~isempty(ind)
            x(j,:) = median(data(ind,:));
        end
    end

    % plot new centroids
    plot(x(1,1),x(1,2),'b^',...
        x(2,1),x(2,2),'r^',...
        x(3,1),x(3,2),'g^');
    pause

    % update clusters
    for i = 1 : l
        d = inf;
        for j = 1 : k
            if norm(data(i,:)-x(j,:),1) < d
                d = norm(data(i,:)-x(j,:),1);
                cluster(i) = j;
            end
        end
    end

    % plot cluster
    c1 = data(cluster==1,:);
    c2 = data(cluster==2,:);
    c3 = data(cluster==3,:);
    plot(c1(:,1),c1(:,2),'bo',c2(:,1),c2(:,2),'ro',c3(:,1),c3(:,2),'go');

    % update objective function
    vnew = 0;
    for i = 1 : l
        vnew = vnew + norm(data(i,:)-x(cluster(i),:),1);
    end
    title(['k-median algoritm: objective function = ',num2str(vnew)]);
    pause

    % stopping criterion
    if v - vnew < 1e-5
        break
    else
        v = vnew;
    end

end
v
```

UNCONSTRAINED OPT. PROBLEMS

1) Gradient method

Gradient method with the Armijo inexact line search

Set $\alpha, \gamma \in (0,1)$, $\bar{t} > 0$. Choose $x^0 \in \mathbb{R}^n$, set $k = 0$.

while $\nabla f(x^k) \neq 0$ **do**
  $d^k = -\nabla f(x^k)$
  $t_k = \bar{t}$
  while $f(x^k + t_k d^k) > f(x^k) + \alpha\, t_k\, (d^k)^\mathsf{T} \nabla f(x^k)$ **do**
    $t_k = \gamma t_k$
  end
  $x^{k+1} = x^k + t_k d$, $k = k + 1$
end

```
alpha = 0.1;
gamma = 0.9;
tbar = 1;
x0 = [ 0 ; 0];
tolerance = 1e-5 ;

%% method

fprintf('Gradient method with Armijo inexact line search\n\n');
fprintf('iter \t f(x) \t\t ||grad f(x)||\n\n');

iter = 0 ;
x = x0 ;

while true
    [v, g] = f(x);
    fprintf('%2.0f \t %1.9f \t %1.7f\n',iter,v,norm(g));

    % stopping criterion
    if norm(g) < tolerance
        break
    end

    % search direction
    d = -g;

    % Armijo inexact line search
    t = tbar ;
    while f(x+t*d) > v + alpha*g'*d*t
        t = gamma*t ;
    end

    % new point
    x = x + t*d ;
    iter = iter + 1 ;
end
```

2) Conjugate gradient method

Conjugate gradient method (quadratic functions)

Choose $x^0 \in \mathbb{R}^n$, set $g^0 = Q x^0 + c$, $k := 0$

while $g^k \neq 0$ **do**

  if $k = 0$ **then** $d^k = -g^k$

  else $\beta_k = \dfrac{(g^k)^\top Q d^{k-1}}{(d^{k-1})^\top Q d^{k-1}}$, $\quad d^k = -g^k + \beta_k d^{k-1}$ $\qquad \beta_k = \dfrac{\|g^k\|^2}{\|g^{k-1}\|^2}$

  end

  $t_k = -\dfrac{(g^k)^\top d^k}{(d^k)^\top Q d^k}$ $\qquad t_k = \dfrac{\|g^k\|^2}{(d^k)^\top Q d^k}$

  $x^{k+1} = x^k + t_k d^k$, $g^{k+1} = Q x^{k+1} + c$, $k = k + 1$

end

```
Q = [6      0     -4      0
     0      6      0     -4
    -4      0      6      0
     0     -4      0      6];
c = [ 1 -1  2 -3 ]';
x0 = [ 10 0 -10 3 ]';
tolerance = 1e-6 ;

%% method

fprintf('Conjugate Gradient method\n\n');
fprintf('iter \t f(x) \t\t ||grad f(x)||\n\n');

iter = 0;

% starting point
x = x0;

while true
    v = 0.5*x'*Q*x + c'*x;
    g = Q*x + c ;
    fprintf('%1.0f \t %1.4f \t %1.4e\n',iter,v,norm(g));

    % stopping criterion
    if norm(g) < tolerance
        break
    end

    %   search direction
    if iter == 0
        d = -g;
    else
        beta = (norm(g)^2)/(norm(g_prev)^2);
        d = -g + beta*d_prev;
    end

    %   step size
    t = (norm(g)^2)/(d'*Q*d);

    %   new point
    iter = iter + 1;
    x = x + t*d;
    d_prev = d ;
    g_prev = g ;
end
```

Conjugate gradient method (nonlinear functions)

Choose $x^0 \in \mathbb{R}^n$, set $k := 0$

while $\nabla f(x^k) \neq 0$ **do**

  if $k = 0$ **then** $d^k = -\nabla f(x^k)$

  else $\beta_k = \dfrac{\|\nabla f(x^k)\|^2}{\|\nabla f(x^{k-1})\|^2}, \quad d^k = -\nabla f(x^k) + \beta_k\, d^{k-1}$

  end

  Compute the step size t_k

  $x^{k+1} = x^k + t_k\, d^k$, $k = k + 1$

end

3) Newton method

Newton method with line search

Set $\alpha, \gamma \in (0,1)$, $\bar{t} > 0$. Choose $x^0 \in \mathbb{R}^n$, set $k = 0$

while $\nabla f(x^k) \neq 0$ **do**

  Solve the linear system $\nabla^2 f(x^k) d^k = -\nabla f(x^k)$

  $t_k = \bar{t}$

  while $f(x^k + t_k\, d^k) > f(x^k) + \alpha\, t_k\, (d^k)^{\mathsf{T}} \nabla f(x^k)$ **do**

    $t_k = \gamma t_k$

  end

  $x^{k+1} = x^k + t_k\, d^k$, $k = k + 1$

end

```
alpha = 0.1;
gamma = 0.9;
tbar = 1;
x0 = [ 0 0 ]';
tolerance = 1e-3 ;
%% method
fprintf('Newton method with line search\n\n');
fprintf('iter \t f(x) \t\t ||grad f(x)||\n\n');
iter = 0 ;
x = x0 ;

while true
    [v, g, H] = f(x);
    fprintf('%1.0f \t %1.7f \t %1.4e\n',iter,v,norm(g));

    % stopping criterion
    if norm(g) < tolerance
        break
    end

    % search direction
    d = -H\g;

    % Armijo inexact line search
    t = tbar ;
    while f(x+t*d) > v + alpha*g'*d*t
        t = gamma*t ;
    end

    % new point
    x = x + t*d ;
    iter = iter + 1 ;

end
```

Quasi-Newton method

Choose $x^0 \in \mathbb{R}^n$, a positive definite matrix H_0, $k = 0$

while $\nabla f(x^k) \neq 0$ **do**
  $d^k = -H_k \nabla f(x^k)$
  Compute step size t_k
  $x^{k+1} = x^k + t_k d^k$, update H_{k+1}, $k = k + 1$
end

How to update matrix H_k?

Davidon-Fletcher-Powell (DFP) method:

$$H_{k+1} = H_k + \frac{p^k (p^k)^{\mathsf{T}}}{(p^k)^{\mathsf{T}} g^k} - \frac{H_k g^k (g^k)^{\mathsf{T}} H_k}{(g^k)^{\mathsf{T}} H_k g^k},$$

$$H_{k+1} = H_k + \left(1 + \frac{(g^k)^{\mathsf{T}} H_k g^k}{(p^k)^{\mathsf{T}} g^k}\right) \frac{p^k (p^k)^{\mathsf{T}}}{(p^k)^{\mathsf{T}} g^k} - \frac{p^k (g^k)^{\mathsf{T}} H_k + H_k g^k (p^k)^{\mathsf{T}}}{(p^k)^{\mathsf{T}} g^k}.$$

(Broyden–Fletcher–Goldfarb–Shanno (BFGS) method).

4) Derivative free methods

Directional direct-search method

Choose starting point $x^0 \in \mathbb{R}^n$, step size $t_0 > 0$, $\beta \in (0, 1)$, tolerance $\varepsilon > 0$ and a positive basis D. Set $k = 0$.

while $t_k > \varepsilon$ **do**
  Order the poll set $\{x^k + t_k d, \quad d \in D\}$
  Evaluate f at the poll points following the chosen order
  If there is a poll point s.t. $f(x^k + t_k d) < f(x^k)$
    then $x^{k+1} = x^k + t_k d$, $t_{k+1} = t_k$ (successful iteration)
  else $x^{k+1} = x^k$, $t_{k+1} = \beta t_k$ (step size reduction)
  end
  $k = k + 1$
end

```
x0 = [ 0 ; 0 ];
t0 = 5 ;
beta = 0.5 ;
epsilon = 1e-5 ;
D = [ 1 0 -1 0 ;
      0 1 0 -1 ] ;
%% method
fprintf('Directional direct-search method\n\n');
fprintf(' x(1) \t\t x(2) \t\t f(x) \n\n');
x = x0;
t = t0;
v = f(x) ;
fprintf('%1.6f \t %1.6f \t %1.6f\n',x(1),x(2),v);
plot(x(1), x(2),'r.');
axis([-6 6 -6 6])
hold on ;
pause
iter = 0;
while t > epsilon
    iter = iter + 1 ;
    newv = v ;
    i = 0 ;
    while (newv >= v) && (i < size(D,2))
        i = i + 1 ;
        newx = x+t*D(:,i);
```

```
        newv = f(newx) ;
        if newv >= v
            plot(newx(1),newx(2),'bs');
            pause
        end
    end
    if newv < v
        x = newx ;
        v = newv ;
        plot(x(1), x(2),'r.');
        fprintf('%1.6f \t %1.6f \t %1.6f\n',x(1),x(2),v);
        pause
    else
        t = beta*t ;
    end
end
```

CONSTRAINED OPT. PROBLEMS

1) Active-set method

Active-set method

0. Choose a feasible point x^0, set $W_0 = \{i : A_i x^0 = b_i\}$ (working set) and $k = 0$.
1. Find the optimal solution y^k of the problem
$$\begin{cases} \min \ \frac{1}{2} x^\mathsf{T} Q x + c^\mathsf{T} x \\ A_i x = b_i \qquad \forall\ i \in W_k \end{cases}$$
2. **If** $y^k \neq x^k$ **then** go to step 3
else go to step 4
3. **If** y^k is feasible **then** $t_k = 1$
else $t_k = \min \left\{ \dfrac{b_i - A_i x^k}{A_i(y^k - x^k)} : \ i \notin W_k, \ A_i(y^k - x^k) > 0 \right\}$,
end
$x^{k+1} = x^k + t_k(y^k - x^k)$, $W_{k+1} = W_k \cup \{i \notin W_k : \ A_i x^{k+1} = b_i\}$,
$k = k+1$ and go to step 1
4. Compute the KKT multipliers μ^k related to y^k
If $\mu^k \geq 0$ **then** STOP
else $x^{k+1} = x^k$, $\mu_j^k = \min\limits_{i \in W_k} \mu_i^k$, $W_{k+1} = W_k \setminus \{j\}$, $k = k+1$ and go to step 1

2) Penalty method

Consider a constrained optimization problem

$$\begin{cases} \min \ f(x) \\ g_i(x) \leq 0 \qquad \forall\ i = 1, \ldots, m \end{cases} \qquad (P)$$

Define the quadratic penalty function

$$p(x) = \sum_{i=1}^{m} (\max\{0, g_i(x)\})^2$$

and consider the unconstrained penalized problem

$$\begin{cases} \min \ f(x) + \dfrac{1}{\varepsilon} p(x) := p_\varepsilon(x) \\ x \in \mathbb{R}^n \end{cases} \qquad (P_\varepsilon)$$

Note that

$$p_\varepsilon(x) \begin{cases} = f(x) & \text{if } x \in \Omega \\ > f(x) & \text{if } x \notin \Omega \end{cases}$$

Penalty method

0. Set $\varepsilon_0 > 0$, $\tau \in (0,1)$, $k = 0$
1. Find an optimal solution x^k of the penalized problem (P_{ε_k})
2. **If** $x^k \in \Omega$ **then** STOP
else $\varepsilon_{k+1} = \tau \varepsilon_k$, $k = k+1$ and go to step 1.

```
%% data
global Q c A b eps;
%% data

Q = [ 1 0 ; 0 2 ] ;
c = [ -3 ; -4 ] ;
A = [-2 1 ; 1 1 ; 0 -1 ];
b = [ 0 ; 4 ; 0 ];
tau = 0.5;
```

```
eps0 = 5;
tolerance = 1e-3;
%% method
fprintf('Penalty method\n\n');
fprintf('eps \t\t x(1) \t\t x(2) \t\t max(Ax-b)\n\n');

options = optimoptions('fminunc','GradObj','on',...
    'Algorithm','quasi-newton','Display','off');

eps = eps0;
while true
    x = fminunc(@p_eps,[0;0],options);
    infeas = max(A*x-b);
    fprintf('%1.2e \t %1.6f \t %1.6f \t %1.3e\n',eps,x(1),x(2),infeas);
    if infeas < tolerance
        break
    else
        eps = tau*eps;
    end
end
%% penalized function

function [v,g] = p_eps(x)

    global Q c A b eps;

    v = 0.5*x'*Q*x + c'*x ;
    g = Q*x + c ;

    for i = 1 : size(A,1)
        v = v + (1/eps)*(max(0,A(i,:)*x-b(i)))^2 ;
        g = g + (2/eps)*(max(0,A(i,:)*x-b(i)))*A(i,:)';
    end

end
```

Consider a convex constrained problem

$$\begin{cases} \min \ f(x) \\ g_i(x) \le 0 \qquad \forall\ i = 1, \dots, m \end{cases} \qquad (P)$$

and define the linear penalty function

$$\tilde{p}(x) = \sum_{i=1}^{m} \max\{0, g_i(x)\}.$$

Then the penalized problem

$$\begin{cases} \min \ f(x) + \frac{1}{\varepsilon}\,\tilde{p}(x) \\ x \in \mathbb{R}^n \end{cases} \qquad (\widetilde{P}_\varepsilon)$$

is unconstrained, convex and nonsmooth.

Exact penalty method

0. Set $\varepsilon_0 > 0$, $\tau \in (0, 1)$, $k = 0$
1. Find an optimal solution x^k of the penalized problem $(\widetilde{P}_{\varepsilon_k})$
2. **If** $x^k \in \Omega$ **then** STOP
   else $\varepsilon_{k+1} = \tau\varepsilon_k$, $k = k + 1$ and go to step 1.

3) Barrier methods

The constrained problem

$$\begin{cases} \min \ f(x) \\ g(x) \leq 0 \end{cases}$$

is equivalent to the unconstrained problem

$$\begin{cases} \min \ f(x) + \sum_{i=1}^{m} I_{-}(g_i(x)) \\ x \in \mathbb{R}^n \end{cases}$$

where

$$I_{-}(u) = \begin{cases} 0 & \text{if } u \leq 0 \\ +\infty & \text{if } u > 0 \end{cases}$$

is called the indicator function of $\mathbb{R}_{-}$, that is neither finite nor differentiable.

The indicator function I_{-} can be approximated by the smooth convex function

$$u \mapsto -\varepsilon \log(-u), \qquad \text{with } \varepsilon > 0,$$

and the approximation improves as $\varepsilon \to 0$.

Hence, we can approximate the problem

$$\begin{cases} \min \ f(x) + \sum_{i=1}^{m} I_{-}(g_i(x)) \\ x \in \mathbb{R}^n \end{cases}$$

with

$$\begin{cases} \min \ f(x) - \varepsilon \sum_{i=1}^{m} log(-g_i(x)) \\ x \in \text{int}(\Omega) \end{cases}$$

$$B(x) = -\sum_{i=1}^{m} log(-g_i(x))$$

is called logarithmic barrier function. It has the following properties:

- $\text{dom}(B) = \text{int}(\Omega)$
- B is convex
- B is smooth with

$$\nabla B(x) = -\sum_{i=1}^{m} \frac{1}{g_i(x)} \nabla g_i(x)$$

$$\nabla^2 B(x) = \sum_{i=1}^{m} \frac{1}{g_i(x)^2} \nabla g_i(x) \nabla g_i(x)^T + \sum_{i=1}^{m} \frac{1}{-g_i(x)} \nabla^2 g_i(x)$$

Logarithmic barrier method

0. Set tolerance $\delta > 0$, $\tau < 1$ and $\varepsilon_1 > 0$. Choose $x^0 \in \text{int}(\Omega)$, set $k = 1$
1. Find the optimal solution x^k of

$$\begin{cases} \min \ f(x) - \varepsilon_k \sum_{i=1}^{m} log(-g_i(x)) \\ x \in \text{int}(\Omega) \end{cases}$$

   using x^{k-1} as starting point
2. **If** $m\,\varepsilon_k < \delta$ **then** STOP
   else $\varepsilon_{k+1} = \tau \varepsilon_k$, $k = k + 1$ and go to step 1

```
global Q c A b eps;

Q = [ 1 0 ; 0 2 ] ;
c = [ -3 ; -4 ] ;
A = [-2 1 ; 1 1 ; 0 -1 ];
b = [ 0 ; 4 ; 0 ];

delta = 1e-3 ;
tau = 0.5 ;
eps1 = 1 ;
x0 = [ 1 ; 1 ];

%% method

fprintf('Logarithmic barrier method\n\n');
fprintf('eps \t\t x(1) \t\t x(2) \t\t gap \n\n');

options = optimoptions('fminunc','GradObj','on',...
    'Algorithm','quasi-newton','Display','off');

x = x0;
eps = eps1 ;
m = size(A,1) ;

while true
    x = fminunc(@logbar,x,options);
    gap = m*eps;
    fprintf('%1.2e \t %1.6f \t %1.6f \t %1.2e\n',eps,x(1),x(2),gap);
    if gap < delta
        break
    else
        eps = eps*tau;
    end
end

%% logarithmic barrier function

function [v,g] = logbar(x)

    global Q c A b eps

    v = 0.5*x'*Q*x + c'*x ;
    g = Q*x + c ;

    for i = 1 : length(b)
        v = v - eps*log(b(i)-A(i,:)*x) ;
        g = g + (eps/(b(i)-A(i,:)*x))*A(i,:)' ;
    end

end
```

MULTIOBJECTIVE

Definition Given a subset $A \subseteq \mathbb{R}^p$, we say

- $x \in A$ is a Pareto ideal minimum (or ideal efficient point) of A if $y \geq x$ for any $y \in A$.
- $x \in A$ is a Pareto minimum (or efficient point) of A if there is no $y \in A$, $y \neq x$ such that $x \geq y$.
- $x \in A$ is a Pareto weak minimum (or weakly efficient point) of A if there is no $y \in A$, $y \neq x$ such that $x > y$, i.e., $x_i > y_i$ for any $i = 1, \ldots, p$.

Given a multiobjective optimization problem

$$\begin{cases} \min \ f(x) = (f_1(x), f_2(x), \ldots, f_p(x)) \\ x \in \Omega \end{cases} \qquad \text{(P)}$$

- $x^* \in \Omega$ is a Pareto ideal minimum of (P) if $f(x^*)$ is an Pareto ideal minimum of $f(\Omega)$, i.e., $f(x) \geq f(x^*)$ for any $x \in \Omega$.
- $x^* \in \Omega$ is a Pareto minimum of (P) if $f(x^*)$ is a Pareto minimum of $f(\Omega)$, i.e., if there is no $x \in \Omega$ such that

$$\begin{array}{ll} f_i(x^*) \geq f_i(x) & \text{for any } i = 1, \ldots, p, \\ f_j(x^*) > f_j(x) & \text{for some } j \in \{1, \ldots, p\}. \end{array}$$

- $x^* \in \Omega$ is a Pareto weak minimum of (P) if $f(x^*)$ is a Pareto weak minimum of $f(\Omega)$, i.e., if there is no $x \in \Omega$ such that

$$f_i(x^*) > f_i(x) \qquad \text{for any } i = 1, \ldots, p.$$

Nel caso di funzioni e constraints lineari possiamo utilizzare questi due teoremi:

Theorem
$x^* \in \Omega$ is a minimum of (P) if and only if the auxiliary optimization problem

$$\begin{cases} \max \ \sum_{i=1}^{p} \varepsilon_i \\ f_i(x) + \varepsilon_i \leq f_i(x^*) & \forall \ i = 1, \ldots, p \\ x \in \Omega \\ \varepsilon \geq 0 \end{cases}$$

has optimal value equal to 0.

Theorem
$x^* \in \Omega$ is a weak minimum of (P) if and only if the auxiliary optimization problem

$$\begin{cases} \max \ v \\ v \leq \varepsilon_i & \forall \ i = 1, \ldots, p \\ f_i(x) + \varepsilon_i \leq f_i(x^*) & \forall \ i = 1, \ldots, p \\ x \in \Omega \\ \varepsilon \geq 0 \end{cases}$$

has optimal value equal to 0.

```
C = [ 1   2 -3 ;
     -1  -1 -1 ;
     -4  -2  1 ];

A = [ 1   1   1 ;
      0   0   1 ;
      -eye(3) ];
```

```
b = [ 10 ; 5 ; 0 ; 0 ; 0 ] ;

% given point

% y = [ 5 ; 0 ; 5 ] ;
% y = [ 4 ; 4 ; 2 ] ;
y = [ 1 ; 4 ; 4 ] ;

%% solve the problem

n = size(C,2);
p = size(C,1);
m = size(A,1);

% check if y is a minimum

c = [zeros(n,1) ; -ones(p,1)] ;
P = [C eye(p);
    A zeros(m,p) ;
    zeros(n,n) -eye(p)] ;
q = [C*y ; b ; zeros(p,1)] ;
options = optimset('Display','off');
[~,v_minimum] = linprog(c,P,q,[],[],[],[],[],options)

% check if y is a weak minimum

c = [ zeros(n,1) ; zeros(p,1) ; -1 ] ;
P = [zeros(p,n) -eye(p) ones(p,1) ;
    C eye(p) zeros(p,1) ;
    A zeros(m,p) zeros(m,1);
    zeros(n,n) -eye(p) zeros(p,1) ] ;
q = [zeros(p,1) ; C*y ; b ; zeros(p,1)] ;
[~,v_weak_minimum] = linprog(c,P,q,[],[],[],[],[],options)
```

First-order optimality conditions: unconstrained problems

Consider an unconstrained multiobjective problem

$$\begin{cases} \min \ f(x) = (f_1(x), f_2(x), \dots, f_p(x)) \\ x \in \mathbb{R}^n \end{cases} \tag{P}$$

where f_i is continuously differentiable for any $i = 1, \dots, p$.

Necessary optimality condition

If x^* is a weak minimum of (P), then there exists $\xi^* \in \mathbb{R}^p$ such that

$$\begin{cases} \sum_{i=1}^{p} \xi_i^* \nabla f_i(x^*) = 0 \\ \xi^* \geq 0, \quad \sum_{i=1}^{p} \xi_i^* = 1 \end{cases} \tag{S}$$

Sufficient optimality condition

If the problem (P) is convex, i.e., f_i is convex for any $i = 1, \dots, p$,
and (x^*, ξ^*) is a solution of the system (S), then x^* is a weak minimum of (P).

First-order optimality conditions: constrained problems

Consider a constrained multiobjective problem

$$\begin{cases} \min\ f(x) = (f_1(x), f_2(x), \ldots, f_p(x)) \\ g_j(x) \le 0 \qquad \forall\ j = 1, \ldots, m \\ h_k(x) = 0 \qquad \forall\ k = 1, \ldots, q \end{cases} \tag{P}$$

where f_i, g_j and h_k are continuously differentiable for any i, j, k.

Necessary optimality condition

If x^* is a weak minimum of (P) and ACQ holds at x^*, then there exist $\xi^* \in \mathbb{R}^p$, $\lambda^* \in \mathbb{R}^m$ and $\mu^* \in \mathbb{R}^q$ such that $(x^*, \xi^*, \lambda^*, \mu^*)$ solves the KKT system

$$\begin{cases} \sum_{i=1}^{p} \xi_i^* \nabla f_i(x^*) + \sum_{j=1}^{m} \lambda_j^* \nabla g_j(x^*) + \sum_{k=1}^{q} \mu_k^* \nabla h_k(x^*) = 0 \\ \xi^* \ge 0, \quad \sum_{i=1}^{p} \xi_i^* = 1 \\ \lambda^* \ge 0 \\ \lambda_j^* g_j(x^*) = 0 \qquad \forall\ j = 1, \ldots, m \end{cases}$$

Sufficient optimality condition

If (P) is convex, i.e., f_i convex, g_j convex and h_k affine, and $(x^*, \xi^*, \lambda^*, \mu^*)$ solves the KKT system, then x^* is a weak minimum of (P).

Scalarization method

Define a vector of weights associated to the objectives:

$$\alpha = (\alpha_1, \ldots, \alpha_p) \ge 0 \quad \text{such that} \quad \sum_{i=1}^{p} \alpha_i = 1$$

and consider the following scalar optimization problem

$$\begin{cases} \min\ \sum_{i=1}^{p} \alpha_i f_i(x) \\ x \in \Omega \end{cases} \tag{P$_\alpha$}$$

Let S_α be the set of optimal solutions of (P$_\alpha$).

Theorem

- $\bigcup_{\alpha \ge 0} S_\alpha \subseteq \{$ weak minima of (P) $\}$
- $\bigcup_{\alpha > 0} S_\alpha \subseteq \{$ minima of (P) $\}$
- If $\alpha \ge 0$ and x^* is the unique optimal solution of (P$_\alpha$), then x^* is a minimum of (P).

```
A = [ 0 -1 ;
     -2  1 ;
      2  1 ];
b = [ 0 ; 0 ; 4 ] ;

%% plot the feasible region

plot([0 2],[0 0],'-k');
plot([2 1],[0 2],'-k');
plot([1 0],[2 0],'-k');

%% solve the scalarized problem with 0 < alfa < 1

options = optimset('Display','off');
for alfa = 0.01 : 0.01 : 0.99
    x = quadprog([2 0 ; 0 2],[8*alfa-6 ; -4],A,b,...
        [],[],[],[],[],options) ;
```

```
    plot(x(1),x(2),'go');
end

%% solve the scalarized problem with alfa = 0

x = quadprog([2 0 ; 0 2],[-6 ; -4],A,b,...
        [],[],[],[],[],options)
plot(x(1),x(2),'ro');

%% solve the scalarized problem with alfa = 1

x = quadprog([2 0 ; 0 2],[2 ; -4],A,b,...
        [],[],[],[],[],options)
plot(x(1),x(2),'bo');
```

Goal method

In the objective space $\mathbb{R}^p$ define the ideal point z as

$$z_i = \min_{x \in \Omega} f_i(x), \qquad \forall\ i = 1, \ldots, p.$$

Since very often (P) has no ideal minimum, i.e., $z \notin f(\Omega)$, we want to find the point of $f(\Omega)$ which is as close as possible to z:

$$\begin{cases} \min \ \|f(x) - z\|_s \\ x \in \Omega \end{cases} \qquad \text{with} \ \ s \in [1, +\infty]. \tag{G}$$

Theorem

- If $s \in [1, +\infty)$, then any optimal solution of (G) is a minimum of (P).
- If $s = +\infty$, then any optimal solution of (G) is a weak minimum of (P).

Assume that (P) is a linear multiobjective optimization problem, i.e.,

$$\begin{cases} \min \ Cx \\ Ax \leq b \end{cases} \tag{P}$$

where C is a $p \times n$ matrix.

If $s = 2$, then (G) is equivalent to a quadratic programming problem:

$$\begin{cases} \min \ \frac{1}{2}\|Cx - z\|_2^2 = \frac{1}{2}x^{\mathsf{T}}C^{\mathsf{T}}Cx - x^{\mathsf{T}}C^{\mathsf{T}}z + \frac{1}{2}z^{\mathsf{T}}z \\ Ax \leq b \end{cases} \tag{G$_2$}$$

If $s = 1$, then (G) is equivalent to the linear programming problem

$$\begin{cases} \min\limits_{x,y} \ \sum\limits_{i=1}^{p} y_i \\ y_i \geq C_i x - z_i & \forall\ i = 1, \ldots, p \\ y_i \geq z_i - C_i x & \forall\ i = 1, \ldots, p \\ Ax \leq b \end{cases} \tag{G$_1$}$$

If $s = +\infty$, then (G) is equivalent to the linear programming problem

$$\begin{cases} \min\limits_{x,y} \ y \\ y \geq C_i x - z_i & \forall\ i = 1, \ldots, p \\ y \geq z_i - C_i x & \forall\ i = 1, \ldots, p \\ Ax \leq b \end{cases} \tag{G$_\infty$}$$

```
C = [ 1  2 -3 ;
     -1 -1 -1 ;
     -4 -2  1 ];

A = [ 1  1  1 ;
      0  0  1
      -eye(3) ];

b = [ 10 ; 5 ; 0 ; 0 ; 0 ] ;

%% ideal point

p = size(C,1);
n = size(C,2);
m = size(A,1);
options = optimset('Display','off');

z = zeros(p,1) ;
for i = 1 : p
    [~,z(i)] = linprog(C(i,:)',A,b,[],[],[],[],[],options);
end
z

%% goal method

% 1-norm

gm1 = linprog([zeros(n,1);ones(p,1)], [C -eye(p); -C -eye(p); A zeros(m,p)],[z;-
z;b],...
    [],[],[],[],[],options);
gm1 = gm1(1:n)

% 2-norm

gm2 = quadprog(C'*C,-C'*z,A,b,[],[],[],[],[],options)

% inf-norm

[gminf, vinf] = linprog([zeros(n,1);1], [C -ones(p,1); -C -ones(p,1); A
zeros(m,1)],[z;-z;b],...
    [],[],[],[],[],options);
gminf = gminf(1:n)


% check if gminf is a minimum

c = [zeros(n,1) ; -ones(p,1)] ;
P = [C eye(p);
     A zeros(m,p) ;
     zeros(n,n) -eye(p)] ;
q = [C*gminf ; b ; zeros(p,1)] ;
[~,v_minimum] = linprog(c,P,q,[],[],[],[],[],options)
```

GAME THEORY

From now on, we will consider noncooperative games with 2 players:

$$\text{Player 1: } \begin{cases} \min \ f_1(x, y) \\ x \in X \end{cases} \qquad \text{Player 2: } \begin{cases} \min \ f_2(x, y) \\ y \in Y \end{cases}$$

How to define an *equilibrium* notion?

Definition

In a two players noncooperative game, a pair of strategies $(\bar{x}, \bar{y})$ is a Nash equilibrium if no player can decrease his/her cost by unilateral deviation, i.e.,

$$f_1(\bar{x}, \bar{y}) = \min_{x \in X} \ f_1(x, \bar{y}), \qquad f_2(\bar{x}, \bar{y}) = \min_{y \in Y} \ f_2(\bar{x}, y).$$

Equivalent definition: $\bar{x}$ is the best response of player 1 to strategy $\bar{y}$ of player 2 and $\bar{y}$ is the best response of player 2 to strategy $\bar{x}$ of player 1.

Matrix games

A **matrix game** is a two-person noncooperative game where:

- X and Y are finite sets: $X = \{1, \ldots, m\}$, $Y = \{1, \ldots, n\}$;
- $f_2 = -f_1$ (zero-sum game).

It can be represented by a $m \times n$ matrix C, where c_{ij} is the amount of money player 1 pays to player 2 if player 1 chooses strategy i and player 2 chooses strategy j.

Strictly dominated strategies

Definition

Given a 2 players noncooperative game, a strategy $x \in X$ is strictly dominated by $\tilde{x} \in X$ if

$$f_1(x, y) > f_1(\tilde{x}, y) \qquad \forall \ y \in Y.$$

Similarly, a strategy $y \in Y$ is strictly dominated by $\tilde{y} \in Y$ if

$$f_2(x, y) > f_2(x, \tilde{y}) \qquad \forall \ x \in X.$$

Strictly dominated strategies can be deleted from the game.

Mixed Strategies

Definition

If C is a $m \times n$ matrix game, then a mixed strategy for player 1 is a m-vector of probabilities and we consider $X = \{x \in \mathbb{R}^m : \ x \geq 0, \quad \sum_{i=1}^m x_i = 1\}$ the set of mixed strategies of player 1.

The vertices of X, i.e., $e_i = (0, \ldots, 0, 1, 0, \ldots, 0)$ are pure strategies of player 1.

$Y = \{y \in \mathbb{R}^n : \ y \geq 0, \quad \sum_{j=1}^n y_j = 1\}$ is the set of mixed strategies of player 2.

The expected costs are $f_1(x, y) = x^\mathsf{T} C y$ (player 1), $f_2(x, y) = -x^\mathsf{T} C y$ (player 2).

Mixed strategies Nash equilibria

Definition

If C is a $m \times n$ matrix game, then $(\bar{x}, \bar{y}) \in X \times Y$ is a mixed strategies Nash equilibrium if

$$\max_{y \in Y} \bar{x}^\mathsf{T} C y = \bar{x}^\mathsf{T} C \bar{y} = \min_{x \in X} x^\mathsf{T} C \bar{y},$$

i.e., $(\bar{x}, \bar{y})$ is a saddle point of the function $x^\mathsf{T} C y$.

Theorem

$(\bar{x}, \bar{y})$ is a mixed strategies Nash equilibrium if and only if

$$\begin{cases} \bar{x} \text{ is an optimal solution of } \min\limits_{x \in X} \max\limits_{y \in Y} x^\mathsf{T} C y \\ \bar{y} \text{ is an optimal solution of } \max\limits_{y \in Y} \min\limits_{x \in X} x^\mathsf{T} C y \end{cases}$$

Theorem

1. The problem $\min\limits_{x \in X} \max\limits_{y \in Y} x^\mathsf{T} C y$ is equivalent to the linear programming problem

$$\begin{cases} \min \ v \\ v \geq \sum_{i=1}^{m} c_{ij} x_i \quad \forall\, j = 1, \dots, n \\ x \geq 0, \quad \sum_{i=1}^{m} x_i = 1 \end{cases} \qquad (P_1)$$

2. The problem $\max\limits_{y \in Y} \min\limits_{x \in X} x^\mathsf{T} C y$ is equivalent to the linear programming problem

$$\begin{cases} \max \ w \\ w \leq \sum_{j=1}^{n} c_{ij} y_j \quad \forall\, i = 1, \dots, m \\ y \geq 0, \quad \sum_{j=1}^{n} y_j = 1 \end{cases} \qquad (P_2)$$

3. (P_2) is the dual of (P_1).

Corollary. Any matrix game has at least a mixed strategies Nash equilibrium.

```
C = [ 1  2 3
      3 -1 3
      3  2 1];

%% solve the LP problem

[m,n] = size(C) ;

[sol,v,~,~,lambda] = linprog([zeros(m,1);1],...
    [C' -ones(n,1)], zeros(n,1),...
    [ones(1,m) 0], 1,...
    [zeros(m,1); -inf], []);

% Nash equilibrium

x = sol(1:m)
y = lambda.ineqlin
```

Bimatrix games

A **bimatrix game** is a two-person noncooperative game where:

- the sets of pure strategies are finite, hence the sets of mixed strategies are $X = \{x \in \mathbb{R}^m : \ x \geq 0, \quad \sum_{i=1}^m x_i = 1\}$ and $Y = \{y \in \mathbb{R}^n : \ y \geq 0, \quad \sum_{j=1}^n y_j = 1\}$;
- $f_2 \neq -f_1$ (non-zero-sum game), the cost functions are $f_1(x,y) = x^\mathsf{T} C_1 y$ and $f_2(x,y) = x^\mathsf{T} C_2 y$, where C_1 and C_2 are $m \times n$ matrices.

Theorem (Nash)

Any bimatrix game has at least a mixed strategies Nash equilibrium.

Theorem

If we define the best response mappings $B_1 : Y \to X$ and $B_2 : X \to Y$ as

$$B_1(y) = \left\{ \text{optimal solutions of } \min_{x \in X} \ x^\mathsf{T} C_1 y \right\},$$

$$B_2(x) = \left\{ \text{optimal solutions of } \min_{y \in Y} \ x^\mathsf{T} C_2 y \right\},$$

then $(\bar{x}, \bar{y})$ is a Nash equilibrium if and only if $\bar{x} \in B_1(\bar{y})$ and $\bar{y} \in B_2(\bar{x})$.

Best response mappings

Nash equilibria are given by the <u>intersections</u> of the graphs of the best response mappings B_1 and B_2:

KKT conditions for Nash equilibria

Theorem

$(\bar{x}, \bar{y})$ is a Nash equilibrium if and only if there are $\mu_1, \mu_2 \in \mathbb{R}$ such that

$$\begin{cases} C_1 \bar{y} + \mu_1 e \geq 0 \\ \bar{x} \geq 0, \quad \sum_{i=1}^m \bar{x}_i = 1 \\ \bar{x}_i (C_1 \bar{y} + \mu_1 e)_i = 0 \quad \forall \ i = 1, \dots, m \\ C_2^\mathsf{T} \bar{x} + \mu_2 e \geq 0 \\ \bar{y} \geq 0, \quad \sum_{j=1}^n \bar{y}_j = 1 \\ \bar{y}_j (C_2^\mathsf{T} \bar{x} + \mu_2 e)_j = 0 \quad \forall \ j = 1, \dots, n \end{cases}$$

where $e = (1, 1, \dots, 1)$.

<u>Al posto di usare il best response c'è quest'altro modo che è un risultato del KKT:</u>

Theorem

Assume that $C_1 < 0$ and $C_2 < 0$.

- If $(\bar{x}, \bar{y})$ is a Nash equilibrium then there are $u > 0$, $v > 0$ such that $\tilde{x} = \bar{x}/u$ and $\tilde{y} = \bar{y}/v$ solve the following system:

$$\begin{cases} \tilde{x} \geq 0, \quad C_1 \tilde{y} + e \geq 0, \quad \tilde{x}_i (C_1 \tilde{y} + e)_i = 0 \quad \forall \ i = 1, \dots, m \\ \tilde{y} \geq 0, \quad C_2^\mathsf{T} \tilde{x} + e \geq 0, \quad \tilde{y}_j (C_2^\mathsf{T} \tilde{x} + e)_j = 0 \quad \forall \ j = 1, \dots, n \end{cases} \qquad \text{(S)}$$

- If $(\tilde{x}, \tilde{y})$ solves system (S) with $\tilde{x} \neq 0$ and $\tilde{y} \neq 0$, then $\left(\frac{\tilde{x}}{\sum_{i=1}^m \tilde{x}_i}, \frac{\tilde{y}}{\sum_{j=1}^n \tilde{y}_j} \right)$ is a Nash equilibrium.

Characterization of Nash equilibria

Define the polyhedra

$$P = \left\{ x \in \mathbb{R}^m : \begin{array}{rl} x_i \geq 0 & \forall\, i = 1, \dots, m \\ (C_2^T x + e)_j \geq 0 & \forall\, j = m+1, \dots, m+n \end{array} \right\}$$

$$Q = \left\{ y \in \mathbb{R}^n : \begin{array}{rl} (C_1 y + e)_i \geq 0 & \forall\, i = 1, \dots, m \\ y_j \geq 0 & \forall\, j = m+1, \dots, m+n \end{array} \right\}$$

Theorem

- $(\tilde{x}, \tilde{y})$ solves system (S) if and only if $\tilde{x} \in P$, $\tilde{y} \in Q$ and for any $k \in \{1, \dots, m+n\}$ either the k-th constraint of P is active in $\tilde{x}$ or the k-th constraint of Q is active in $\tilde{y}$.
- If the vertices of P and Q are non-degenerate and $(\tilde{x}, \tilde{y})$ solves system (S), then $\tilde{x}$ is a vertex of P and $\tilde{y}$ is a vertex of Q.

Therefore, if $C_1 < 0$, $C_2 < 0$ and vertices of P and Q are non-degenerate, then we can find all the Nash equilibria analyzing all the pairs (x, y) of vertices of P and Q, checking if each constraint $k = 1, \dots, m+n$ is active either in x or in y.

Convex games

Now, we consider a two-person noncooperative game

Player 1: $\begin{cases} \min_x \ f_1(x, y) \\ g_i^1(x) \leq 0 \quad \forall\, i = 1, \dots, p \end{cases}$ Player 2: $\begin{cases} \min_y \ f_2(x, y) \\ g_j^2(y) \leq 0 \quad \forall\, j = 1, \dots, q \end{cases}$

where f_1, g^1, f_2 and g^2 are continuously differentiable.
The game is said convex if the optimization problem of each player is convex.

Theorem

If the feasible regions $X = \{x \in \mathbb{R}^m : \ g_i^1(x) \leq 0 \quad i = 1, \dots, p\}$ and $Y = \{y \in \mathbb{R}^n : \ g_j^2(y) \leq 0 \quad j = 1, \dots, q\}$ are closed, convex and bounded, the cost function $f_1(\cdot, y)$ is quasiconvex for any $y \in Y$ and $f_2(x, \cdot)$ is quasiconvex for any $x \in X$, then there exists at least a Nash equilibrium.

The quasiconvexity of the cost functions is crucial.

Theorem

- If $(\bar{x}, \bar{y})$ is a Nash equilibrium and the Abadie constraints qualification holds both in $\bar{x}$ and $\bar{y}$, then there are $\lambda^1 \in \mathbb{R}^p$, $\lambda^2 \in \mathbb{R}^q$ such that

$$\begin{cases} \nabla_x f_1(\bar{x}, \bar{y}) + \sum_{i=1}^{p} \lambda_i^1 \nabla g_i^1(\bar{x}) = 0 \\ \lambda^1 \geq 0, \qquad g^1(\bar{x}) \leq 0 \\ \lambda_i^1 g_i^1(\bar{x}) = 0, \quad i = 1, \dots, p \\ \nabla_y f_2(\bar{x}, \bar{y}) + \sum_{j=1}^{q} \lambda_j^2 \nabla g_j^2(\bar{y}) = 0 \\ \lambda^2 \geq 0, \qquad g^2(\bar{y}) \leq 0 \\ \lambda_j^2 g_j^2(\bar{y}) = 0, \quad j = 1, \dots, q \end{cases}$$

- If $(\bar{x}, \bar{y}, \lambda^1, \lambda^2)$ solves the above system and the game is convex, then $(\bar{x}, \bar{y})$ is a Nash equilibrium.

Merit functions

Merit functions allow reformulating the Nash equilibrium problem into an equivalent optimization problem.

Assume that the game is convex. Consider the Nikaido-Isoda function

$$f(x,y,u,v) = f_1(u,y) - f_1(x,y) + f_2(x,v) - f_2(x,y),$$

where $x,u \in \mathbb{R}^m$ and $y,v \in \mathbb{R}^n$. Define the gap function as

$$\psi(x,y) = \max_{u\in X, v\in Y} [-f(x,y,u,v)].$$

Then:

- The problem defining ψ is convex
- $\psi(x,y) \geq 0$ for any $(x,y) \in X \times Y$
- $(\bar{x},\bar{y})$ is a Nash equilibrium if and only if $(\bar{x},\bar{y}) \in X \times Y$ and $\psi(\bar{x},\bar{y}) = 0$

Therefore, finding Nash equilibria is equivalent to solve the constrained optimization problem

$$\begin{cases} \min\ \psi(x,y) \\ (x,y) \in X \times Y \end{cases}$$

In general ψ is not differentiable, but it is possible to regularize it.

Given a parameter $\alpha > 0$, the regularized gap function is defined as

$$\psi_\alpha(x,y) = \max_{u\in X, v\in Y} \left[-f(x,y,u,v) - \frac{\alpha}{2}\|(x,y)-(u,v)\|^2\right].$$

Then:

- The problem defining ψ_α is convex and has a unique optimal solution
- ψ_α is continuously differentiable
- $\psi_\alpha(x,y) \geq 0$ for any $(x,y) \in X \times Y$
- $(\bar{x},\bar{y})$ is a Nash equilibrium if and only if $(\bar{x},\bar{y}) \in X \times Y$ and $\psi_\alpha(\bar{x},\bar{y}) = 0$.

Therefore, finding Nash equilibria is equivalent to solve the smooth constrained optimization problem

$$\begin{cases} \min\ \psi_\alpha(x,y) \\ (x,y) \in X \times Y \end{cases}$$

It is possible to reformulate the problem of finding Nash equilibria as an unconstrained optimization problem.

Given two parameters $\beta > \alpha > 0$, the D-gap function is defined as

$$\psi_{\alpha,\beta}(x,y) = \psi_\alpha(x,y) - \psi_\beta(x,y).$$

Then:

- $\psi_{\alpha,\beta}$ is continuously differentiable
- $\psi_{\alpha,\beta}(x,y) \geq 0$ for any $(x,y) \in \mathbb{R}^m \times \mathbb{R}^n$
- $(\bar{x},\bar{y})$ is a Nash equilibrium if and only if $\psi_{\alpha,\beta}(\bar{x},\bar{y}) = 0$.

Therefore, finding Nash equilibria is equivalent to solve the smooth, unconstrained optimization problem

$$\begin{cases} \min\ \psi_{\alpha,\beta}(x,y) \\ (x,y) \in \mathbb{R}^m \times \mathbb{R}^n \end{cases}$$

```
C1 = [ 3   3
    4  1
    6  0 ];
C2 = [ 3   4
    4  0
    3  5 ] ;
```

```
[m,n] = size(C1) ;
%% check if w is a Nash equilibrium
w = [ 1/3 1/3 1/3 1/2 1/2 ]';
gap(w)
reggap(w,1)
Dgap(w,1,10)
%% find a local minimum of the regularized gap function
fprintf('Regularized gap function - local minimum\n');
alfa = 1 ;
% find a local minimum
options = optimset('Display','off');
[locmin,optval] = fmincon(@(z) reggap(z,alfa),w,[],[],...
    [ones(1,m) zeros(1,n) ; zeros(1,m) ones(1,n)], [1;1],...
    zeros(m+n,1),[],[],options)
%% try to find a global minimum of the regularized gap function
%  with a multistart approach
fprintf('Regularized gap function - multistart approach\n');
for i = 1 : 100

    % starting point
    x0 = rand(m+n,1);
    x0(1:m) = x0(1:m)/sum(x0(1:m));
    x0(m+1:m+n) = x0(m+1:m+n)/sum(x0(m+1:m+n));

    % find a local minimum
    [locmin,optval] = fmincon(@(z) reggap(z,alfa),x0,[],[],...
        [ones(1,m) zeros(1,n) ; zeros(1,m) ones(1,n)],...
        [1;1],zeros(m+n,1),ones(m+n,1),[],options);
    if optval < 1e-4
        locmin
        optval
        break
    end
end
%% try to find a global minimum of the D-gap function
%  with a multistart approach
fprintf('D-gap function - multistart approach\n');

alfa = 1 ;
beta = 10 ;

for i = 1 : 100

    % starting point
    x0 = rand(m+n,1);
    x0(1:m) = x0(1:m)/sum(x0(1:m));
    x0(m+1:m+n) = x0(m+1:m+n)/sum(x0(m+1:m+n));

    % find a local minimum
    [locmin,optval] = fminunc(@(z) Dgap(z,alfa,beta),x0,options);
    if optval < 1e-4
        locmin
        optval
        break
    end
end
```

```
%GAP FUNCTION
function v = gap(z)

global C1 C2

[m,n] = size(C1);

x = z(1:m);
y = z(m+1:m+n);
v = x'*(C1+C2)*y - min(C1*y) - min(C2'*x);

%Altra soluzione
% options = optimset('Display', 'off');
% [~,v1] = linprog(C1*y,[],[],ones(1,m),1,zeros(m,1),[],options);
% [~,v2] = linprog(C2'*y,[],[],ones(1,n),1,zeros(n,1),[],options);
% v = x'*(C1+C2)*y - v1 - v2;

end



%REGULARIZED GAP FUNCTION
function v = reggap(z,alfa)

global C1 C2

[m,n] = size(C1);
x = z(1:m);
y = z(m+1:m+n);
options = optimset('Display','off');
[~,v1] = quadprog(alfa*eye(m),C1*y-alfa*x,[],[],ones(1,m),1,...
    zeros(m,1),ones(m,1),[],options);
[~,v2] = quadprog(alfa*eye(n),C2'*x-alfa*y,[],[],ones(1,n),1,...
    zeros(n,1),ones(n,1),[],options);

v = x'*(C1+C2)*y - 0.5*alfa*(norm(x)^2 + norm(y)^2) - v1 - v2;

end


%DGAP FUNCTION
function v = Dgap(z,alfa,beta)

v = reggap(z,alfa) - reggap(z,beta);

end
```