

Check convexity

Convex \Leftrightarrow autodromo: $\nabla^2 f(x) \geq 0$ cioè positive semidefinite

Strictly convex \Leftrightarrow autodromo: $\nabla^2 f(x) > 0$ cioè positive definite

Strongly convex \Leftrightarrow autodromo: $\nabla^2 f(x) \geq \gamma$ con $\gamma > 0$

OPT. Problem constrained

$$\max(f(x)) = -\min(-f(x))$$

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

Un problema è convex se:

- 1) f è convex
- 2) g_1, \dots, g_m convex
- 3) h_1, \dots, h_p affine (lineari)

Vantaggi:

- 1) Ω convex set
- 2) local opt = global opt

Esercizio global minimum

1) f continua e Ω chiusa e limitata (Weierstrass)

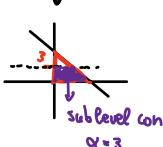
2) f, g_i, h_j continue, D chiuso e Ω limitata

3) f continua, Ω chiusa e $\exists \alpha$. $\underline{\alpha}$ -sublevel set non vuoto e limitato

$$S_\alpha(f) = \{x \in \Omega : f(x) \leq \alpha\}$$

4) f continua e coercive, Ω chiuso

$$\lim_{\|x\| \rightarrow \infty} f(x) = +\infty$$



5) f strongly convex (\Rightarrow coercive) e Ω chiuso

6) f strongly convex e Ω chiusa e convessa \Rightarrow unique global min.

7) Teorema di Eaves (quadratic problems) - Esistenza global min.

Consideriamo il recession cone $\text{rec}(\Omega) = \{d : Ad \leq 0\}$, ed esiste global. min \Leftrightarrow :

a) $d^T Q d \geq 0 \quad \forall d \in \text{Rec}(\Omega)$

b) $d^T (Qx + c) \geq 0 \quad \forall x \in \Omega \text{ e } \forall d \in \text{rec}(\Omega) \text{ t.c. } d^T Qd = 0$

OPT problem unconstrained

$$\min\{f(x) : x \in \mathbb{R}^n\}.$$

se f è convex, allora x^* è un ottimo globale $\Leftrightarrow \nabla f(x^*) = 0$

ACQ (Abadie Constraints Qualifications)

Sufficient conditions per far sì che

ACQ holds $\forall x \in \Omega$

a) Affine Constraints: g_i, h_j affine

b) Slater Condition:

g_i convex, h_j affine e $\exists \bar{x} \in \text{int}(\Omega)$ t.c.

$$g_i(\bar{x}) < 0 \text{ e } h_j(\bar{x}) = 0$$

c) linear indip. of the gradients of active constn.)

$$\text{Se } \bar{x} \in \Omega \text{ e : vettori } \begin{cases} \nabla g_i(\bar{x}) & \text{for } i \in \mathcal{A}(\bar{x}), \\ \nabla h_j(\bar{x}) & \text{for } j = 1, \dots, p \end{cases}$$

sono lin. indipendenti

Theorem

- If f is convex and $\alpha > 0$, then αf is convex
- If f_1 and f_2 are convex, then $f_1 + f_2$ are convex
- If f is convex, then $f(Ax + b)$ is convex

Examples

- If f is convex, then $e^{f(x)}$ is convex
- If f is concave and positive, then $\log f(x)$ is concave
- If f is convex, then $-\log(-f(x))$ is convex on $\{x : f(x) < 0\}$
- If f is concave and positive, then $\frac{1}{f(x)}$ is convex
- If f is convex and nonnegative, then $f(x)^p$ is convex for all $p \geq 1$

Special cases

- $Q = 0$, linear case
- $d^T C > 0 \quad \forall d \in \text{Rec}(\Omega)$
- Q positive definite
- Ω bounded

KKT (Kanush - Kuhn - Tucker theorem)

Se x^* è un ottimo locale, ACO holds in x^* $\Rightarrow \exists \lambda^* \in \mu^* + c$ (x^*, λ^*, μ^*) soddisfa il KKT system

$$\left\{ \begin{array}{l} \nabla f(x) + \sum_{i=1}^m \lambda_i^* \nabla g_i(x) + \sum_{j=1}^p \mu_j^* \nabla h_j(x) = 0 \\ \lambda_i g_i(x) = 0 \\ \lambda \geq 0 \\ g(x) \leq 0 \\ h(x) = 0 \end{array} \right.$$

OSS: Non è detto che le sol. del KKT system siano ottimi locali

Due casi:

- 1) Se il problema è convex e (x^*, λ^*, μ^*) risolve il KKT system $\Rightarrow x^*$ è un ottimo globale
- 2) Se il problema non è convex, le soluzioni (x^*, λ^*, μ^*) non sono ottimi locali.

Per verificarlo:

a) Considero il critical cone: $C(x^*, \lambda^*, \mu^*) = \{d \in D(x^*) : d^T \nabla f(x^*) = 0\}$

b) Considero la lagrangian function: $L(x, \lambda, \mu) := f(x) + \sum_{i=1}^m \lambda_i g_i(x) + \sum_{j=1}^p \mu_j h_j(x)$.

c) $d^T \nabla_x^2 L(x^*, \lambda^*, \mu^*) d > 0 \quad \forall d \in C(x^*, \lambda^*, \mu^*), d \neq 0 \Rightarrow x^* \text{ local opt.}$

\hookrightarrow Nel caso degli unconstrained problems: se $\nabla f(x^*) = 0$ e $\nabla^2 f(x^*)$ positive definite $\Rightarrow x^* \text{ local opt.}$

Lagrangian Dual Problem

$$\max_{\lambda \geq 0} \varphi(\lambda, \mu) \quad \text{dove } \varphi(\lambda, \mu) = \inf_{x \in D} L(x, \lambda, \mu) \text{ is the Lagrangian dual function.}$$

Caratteristiche:

1) Sempre concave

2) siamo trovando il miglior lower bound per $v(p)$ perché $\forall \lambda \geq 0, \mu \in \mathbb{R}^p \quad \varphi(\lambda, \mu) \leq v(p)$

3) weak duality: $v(D) \leq v(P) \Rightarrow$ cioè il valore ottimo per (D) è il miglior lower bound del valore ottimo per (P)

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^m \lambda_i g_i(x) + \sum_{j=1}^p \mu_j h_j(x)$$

4) Strong duality: Se P convex, $\exists x^*$ global min, ACA holds in x^* :

- a) (λ^*, μ^*) associati a x^* sono ottimi globali per D
- b) $v(D) = v(P)$

linear SVM

consiste nel trovare un hyperpiano $(w^T x + b = 0)$ che separa per fallamente i due insiemi A e B , che massimizza

Stiamo cercando l'hyperpiano con il maggiore "margin of separation", cioè la min distanza fra H

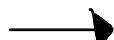
e il training point più vicino.

$$\begin{cases} \min_{w,b} \frac{1}{2} \|w\|^2 \\ w^T x^i + b \geq 1 \quad \forall x^i \in A \\ w^T x^j + b \leq -1 \quad \forall x^j \in B \end{cases}$$



Dual problem

$$\begin{cases} \max_{\lambda} -\frac{1}{2} \lambda^T Q \lambda + c^T \lambda \\ \sum_{i=1}^{\ell} \lambda_i y^i = 0 \\ \lambda \geq 0 \end{cases}$$



dove $X = (y^1 x^1, y^2 x^2, \dots, y^\ell x^\ell)$ and the vector $e^T = (1, \dots, 1)$.

```
% Calculate the number of rows in matrices A and B
nA = size(A,1);
nB = size(B,1);
```

```
% training points
T = [A ; B];
```

```
%% Linear SVM - primal model
```

```
% Define the quadratic coefficient matrix for the optimization problem
Q = [ 1 0 0 ;
      0 1 0 ;
      0 0 0 ];
```

```
% Define the linear coefficient matrix for the optimization problem
D = [-ones(nA,1);
      B ones(nB,1) ] ;
```

```
% Define the constant term for the optimization problem
d = -ones(nA+nB,1) ;
```

```
% Solve the optimization problem using quadprog function
options = optimset('LargeScale','off','Display','off');
sol = quadprog(Q,zeros(3,1),D,d,[],[],[],[],options);
```

```
w = sol(1:2)
```

```
b = sol(3)
```

```
% Number of samples in A and B
nA = size(A,1);
nB = size(B,1);
```

```
% training points
T = [A ; B];
```

```
%% Linear SVM - dual model
```

```
% define the problem
y = ones(nA,1); % labels
l = length(y);
Q = (y * y') .* (T * T'); % Q is a matrix that represents the dot product of each pair of feature vectors, scaled by the labels
```

```
% solve the problem
options = optimset('LargeScale','off','Display','off');
la = quadprog(Q,-ones(l,1),y,0,zeros(l,1),[],[],options);

% calculate the weight vector w
w0 = zeros(2,1);
for i = 1:l;
    w0 = w0 + la(i)*y(i)*T(:,i)';
end
w0
```

```
% calculate the bias term b
```

```
ind = find(la > 1e-3); % find the indices of the non-zero Lagrange multipliers (i.e., the support vectors)
i = ind(1); % choose the first support vector
b0 = 1/y(i) - w0'*T(:,i); % calculate the bias term
```

```
%% support vectors
```

```
supp = find(la > 1e-3); % find the indices of the support vectors
suppA = supp(supp <= nA); % indices of the support vectors in A
suppB = supp(supp > nA) - nA; % indices of the support vectors in B
```

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

$$b^* = \frac{1}{y^i} - (w^*)^T x^i$$

uso x^i, y^i che corrispondono a una $\lambda_i^* > 0$ (ne bastano una)

$$f(x) = \text{sign}((w^*)^T x + b^*).$$

1) Convex quadratic proq. problem

2) Dual constraints più facili

3) $\lambda_i^* > 0 \rightarrow x^i$ support vector

Linear SVM (soft Margin)

$$\max(f(x)) - \min(-f(x))$$

Considero possibili outliers introducendo delle slack variables $\xi_i \geq 0$

$\sum_{i=1}^e \xi_i$ è un limite superiore del numero di outliers, mentre $C > 0$ è un parametro

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

↓ duale

$$\begin{cases} \max_{\lambda} -\frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} [y^i y^j (x^i)^T x^j] \lambda_i \lambda_j + \sum_{i=1}^{\ell} \lambda_i \cdot 1 \\ \sum_{i=1}^{\ell} \lambda_i y^i = 0 \quad \text{eq} \\ 0 \leq \lambda_i \leq C \quad i = 1, \dots, \ell \end{cases}$$



```
% Calculate the number of rows in matrices A and B
nA = size(A,1);
nB = size(B,1);

% training points
T = [A ; B];

%% Linear SVM with soft margin - dual model

% define the problem
C = 10; % Set the regularization parameter C
y = [ones(nA,1); -ones(nB,1)]; % Define the labels for the training set
Q = (y * y') .* (T * T'); % Calculate the kernel matrix for the training set

% Solve the optimization problem using quadprog function
options = optimset('LargeScale','off','display','off');
la = quadprog(Q,-ones(1,1),[],y',0,zeros(1,1),C*ones(1,1),[],options);

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

% Calculate the bias term b
ind = find((la > 1e-3) & (la < C-1e-3));
i = ind(1);
bD = 1/y(ind) - wD'*T(ind,:);

%% support vectors
supp = find(la > 1e-3);
suppA = supp(supp <= nA);
suppB = supp(supp > nA)-nA;
```

Oss: i punti che sono missclassified o che sono all'interno dell'hyperplane hanno $\lambda^* = C$ dovuto alle

complementarity conditions

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

e che la loro $\xi_i \neq 0$

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

$$b^* = \frac{1}{y^i} - (w^*)^T x^i.$$

↓
se $0 < \lambda_i^* < C$

$$f(x) = \text{sign}((w^*)^T x + b^*).$$

Non linear SVM

Se il problema non è lineare possiamo mapparlo in uno spazio di dimensioni maggiori,

tramite una map function $\phi: \mathbb{R}^n \rightarrow H$, nel quale è linearmente separabile.

Primal problem:

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

⇒ Problema: w è un vettore in uno spazio di dimensionalità maggiore, potrei avere infinite variabili.

Duale

V

Dual problem:

$$\begin{cases} \max_{\lambda} -\frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} y^i y^j \phi(x^i)^T \phi(x^j) \lambda_i \lambda_j + \sum_{i=1}^{\ell} \lambda_i \\ \sum_{i=1}^{\ell} \lambda_i y^i = 0 \\ 0 \leq \lambda_i \leq C \quad i = 1, \dots, \ell \end{cases}$$

→ Risolvo il problema (non ho più w) e non devo conoscere specificatamente $\phi(x)$, ma solo il prodotto $\phi(x)^T \cdot \phi(y)$

Utilizzo una Kernel function che rappresenta implicitamente il prodotto $k(x,y) = \langle \phi(x), \phi(y) \rangle$

The dual problem depends on the kernel k :

$$\begin{cases} \max_{\lambda} -\frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} y^i y^j k(x^i, x^j) \lambda_i \lambda_j + \sum_{i=1}^{\ell} \lambda_i \cdot 1 \\ \sum_{i=1}^{\ell} \lambda_i y^i = 0 \\ 0 \leq \lambda_i \leq C \quad i = 1, \dots, \ell \end{cases}$$

% Calculate the number of rows in matrices A and B
nA = size(A,1);
nB = size(B,1);

% Concatenate the rows of A and B to create the training set
T = [A ; B];

% Define the labels for the training set
y = [ones(nA,1) ; -ones(nB,1)];
l = length(y);

% Nonlinear SVM

% Set the regularization parameter C
C = 1;

% Set the value of gamma for the Gaussian kernel
gamma = 1;

% Initialize the kernel matrix
K = zeros(l,l);

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

% Calculate the quadratic coefficient matrix for the optimization 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 optimization problem using quadprog function
options = optimset('LargeScale','off','display','off');

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

% Calculate the bias term b

ind = find((la > 1e-3) & (la < C-1e-3));

i = ind(1);

b = 1/y(i);

for j = 1 : l

b = b - la(j)*y(j)*K(i,j);

end

Examples:

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

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

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

Polynomial Regression

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

Vogliamo trovare il polinomio p , di grado $n-1$, che approssima al meglio i dati, cioè vogliamo trovare i coefficienti z di p t.c la norma del residual vector r , $r_i = p(x_i) - y_i$, è minima.

$$\begin{cases} \min_{z \in \mathbb{R}^n} \|Az - y\| \\ \text{è convessa } \forall \text{ norma} \end{cases}$$

dove $A = \begin{pmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_\ell & x_\ell^2 & \dots & x_\ell^{n-1} \end{pmatrix}$ e $y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_\ell \end{pmatrix}$

$x = \text{data(:,1)}$ → 1° col

$y = \text{data(:,2)}$ → 2° col

$\| \cdot \|_2$

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

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

Essendo $A^T A$ positive definita, $f(x)$ è strettamente convessa.

Quindi il problema è unconstrained, quindi la

soluzione corrisponde a $\nabla f(x) = 0$, cioè

$$\nabla f = A^T Az - A^T y = 0$$

$$A^T Az = A^T y$$

```
% Extract the first and second columns of the data matrix
x = data(:,1);
y = data(:,2);

% Get the number of data points
l = length(x);

% Set the degree of the polynomial
n = 4;

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

%% 2-norm problem

% Solve for the polynomial coefficients using 2-norm regularization
z2 = (A'*A)\(A'*y);

% Calculate the fitted values using the polynomial coefficients
p2 = A*z2;
```

$\| \cdot \|_1$

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

```
% Extract the first and second columns of the data matrix
x = data(:,1);
y = data(:,2);

% Get the number of data points
l = length(x);

% Set the degree of the polynomial
n = 4;

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

%% 1-norm problem

% Set the objective function coefficients
c = [zeros(n,1); ones(l,1)];

% Set the constraint matrix
D = [A -eye(l); -A -eye(l)];

% Set the right-hand side of the constraints
d = [y; -y];

% Solve the optimization problem using linprog
sol1 = linprog(c,D,d);

% Extract the polynomial coefficients from the solution
z1 = sol1(1:n);

% Calculate the fitted values using the polynomial coefficients
p1 = A*z1;
```

$\| \cdot \|_\infty$

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

```
% Extract the first and second columns of the data matrix
x = data(:,1);
y = data(:,2);

% Get the number of data points
l = length(x);

% Set the degree of the polynomial
n = 4;

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

%% inf-norm problem

% Set the objective function coefficients
c = [zeros(n,1); 1];

% Set the constraint matrix
D = [A -ones(l,1); -A -ones(l,1)];

% Solve the optimization problem using linprog
solinf = linprog(c,D,d);

% Extract the polynomial coefficients from the solution
zinf = solinf(1:n);

% Calculate the fitted values using the polynomial coefficients
pinf = A*zinf;
```

Questo problema corrisponde al seguente problema:

$$\begin{cases} \min_u u \\ u \geq A_i z - y_i \quad \forall i = 1, \dots, l \\ u \geq y_i - A_i z \quad \forall i = 1, \dots, l \end{cases}$$

E-SV regression

Vogliamo trovare una funzione f che ha una derivazione E massima dai target y_i e che sia più piatta possibile.

Consideriamo $f(x) = w^T x + b$, E come parmetro e w piccolo indica la flatness, quindi:

$$\begin{cases} \min_{w,b} \frac{1}{2} \|w\|^2 \\ y_i \leq w^T x_i + b + \varepsilon \quad \forall i = 1, \dots, \ell \\ y_i \geq w^T x_i + b - \varepsilon \quad \forall i = 1, \dots, \ell \end{cases}$$

```
% Extract the first and second columns of the data matrix
x = data(:,1);
y = data(:,2);

% Get the number of data points
l = length(x);

% Set the degree of the polynomial
n = 4;

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

%% inf-norm problem

% Set the objective function coefficients
c = [zeros(n,1); 1];

% Set the constraint matrix
D = [A -ones(l,1); -A -ones(l,1)];

% Solve the optimization problem using linprog
solinf = linprog(c,D,d);

% Extract the polynomial coefficients from the solution
zinf = solinf(1:n);

% Calculate the fitted values using the polynomial coefficients
pinf = A*zinf;
```

E-SV regression con slack variables

Se E è troppo piccolo, il modello non potrebbe non essere possibile (ho degli outliers)

Primal Problem (P):

$$\begin{cases} \min_{w, \xi^+, \xi^-} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{\ell} (\xi_i^+ + \xi_i^-) \\ y_i \leq w^T x_i + b + \xi_i^+ \quad \forall i = 1, \dots, \ell \\ y_i \geq w^T x_i + b - \xi_i^- \quad \forall i = 1, \dots, \ell \\ \xi^+ \geq 0 \\ \xi^- \geq 0 \end{cases}$$

```
% Extract the first and second columns of the data matrix
x = data(:,1);
y = data(:,2);

% Get the number of data points
l = length(x);

%% linear regression - primal problem with slack variables

% Set the parameter epsilon
epsilon = 0.2;

% Set the parameter C
C = 10;

% Set the quadratic coefficient matrix
Q = [1 zeros(1, 2*l+1); zeros(2*l+1) zeros(2*l+1)];

% Set the linear coefficient vector
c = [0; 0; C*ones(2*l, 1)];

% Set the constraint matrix
D = [-x -ones(l, 1) -eye(l) zeros(l, 1) x ones(l, 1) zeros(l) -eye(l)];

% Set the right-hand side of the constraints
d = epsilon*ones(2*l, 1) + [-y; y];

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

% Extract the slope and intercept of the linear regression line
w = sol(1);
b = sol(2);

% Extract the slack variables xi+ and xi-
xip = sol(3:2:l);
xim = sol(3:l+2:l);

% Calculate the fitted values and the upper and lower bounds of the epsilon-tube
z = w.*x + b;
zp = w.*x + b + epsilon;
zm = w.*x + b - epsilon;
```

Dual Problem (D):

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

```
% Data
x = data(:, 1);
y = data(:, 2);
n = length(x);

%% linear regression - dual problem

% Parameters
epsilon = 0.2;
C = 10;

% Define the problem
X = x.*x';
Q = [X -X; -X X];
c = epsilon * ones(2 * n, 1) + [-y; y];

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

% Compute w and b
w = (lap - lam)' * x;
if any(lap > 1e-3 & lap < C - 1e-3)
    i = find(lap > 1e-3 & lap < C - 1e-3, 1);
    b = y(i) - w * x(i) - epsilon;
else
    i = find(lam > 1e-3 & lam < C - 1e-3, 1);
    b = y(i) - w * x(i) + epsilon;
end

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

$$w = \sum_{i=1}^{\ell} (\lambda_i^+ - \lambda_i^-) x_i,$$

Hence, if there is some i s.t. $0 < \lambda_i^+ < C$, then $b = y_i - w^T x_i - \varepsilon$; if there is some i s.t. $0 < \lambda_i^- < C$, then $b = y_i - w^T x_i + \varepsilon$.

Non-linear E-SVM

Primal problem:

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

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

↓

Dual

→

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

OSS: Numero di variabili finite = 2ℓ

Examples:

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

► find b :

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

or

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

► Recession function

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

```
% Extract first and second columns of data and store in x and y
x = data(:,1);
y = data(:,2);
% Store the number of points in l
l = length(x);

%% nonlinear regression - dual problem

epsilon = 0.2;
C = 10;

% Initialize the X matrix
X = zeros(l, 1);
% Populate the X matrix with kernel values
for i = 1:l
    for j = 1:l
        X(i,j) = kernel(x(i), x(j));
    end
end
% Create the Q and c matrices for the quadratic program
Q = (X-X); % -X X';
c = epsilon * ones(2 * l, 1) + [-y; y];

% Solve the quadratic program using the quadprog function
sol = quadprog(Q, c, [], [], [ones(1, l), -ones(1, l)], 0, zeros(2 * l, 1), C * ones(2 * l, 1));
% Extract the lagrange multipliers from the solution
lam = sol(1:l);
lap = sol(l+1:l);

% Define the kernel function
function v = kernel(x, y)
    gamma = 1;
    v = exp(-gamma * norm(x - y)^2);
end
```

Clustering Problem

cluster

Vogliamo trovare k subset omogenei e ben separati da un insieme di punti S .

Assegniamo ogni punto al centroide più vicino secondo una metrica che rapp. la distanza:

a) $d(x, y) = \|x - y\|_2^2$

b) $d(x, y) = \|x - y\|_1$

Il problema consiste nel trovare k centroidi che minimizzano la somma delle distanze tra ogni punto e il centroide

più vicino.

$$\begin{cases} \min \sum_{i=1}^{\ell} \min_{j=1, \dots, k} d(p_i, x_j) \\ x_j \in \mathbb{R}^n \quad \forall j = 1, \dots, k \end{cases}$$

K-means ($\| \cdot \|_2$)

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

```

function [x,cluster,v] = kmeans(data,k,InitialCentroids)
% Initialize variables
n = size(data, 1); % number of patterns
x = InitialCentroids; % initialize centroids
cluster = zeros(n, 1); % initialize clusters

% Assign patterns to nearest centroids
for i = 1:n
    d = inf;
    for j = 1:k
        dist = norm(data(i, :) - x(j, :));
        if dist < d
            d = dist;
            cluster(i) = j;
        end
    end
end

% Compute objective function value
v = sum(norm(data - x(cluster, :)) .^ 2);

% Iterate until convergence
while true
    % Update centroids
    for j = 1:k
        ind = find(cluster == j);
        if ~isempty(ind)
            x(j, :) = mean(data(ind, :));
        end
    end

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

    % Update objective function value
    v_new = sum(norm(data - x(cluster, :)) .^ 2);

    % Check for convergence
    if v - v_new < 1e-5
        break
    else
        v = v_new;
    end
end

```

```

k = 3;
% multistart approach
vbest = inf;
xbest = [];
clusterbest = [];
maxiter = 100;
iter = 0;

while iter < maxiter
    InitialCentroids = 10*rand(k,2);
    [x,cluster,v] = kmeans(data,k,InitialCentroids);
    if v < vbest
        xbest = x;
        clusterbest = cluster;
        vbest = v;
    end
    iter = iter + 1;
end
*
```

Oss: 1) è possibile migliorare K-means e K-median eseguendoli più volte inizializzando i centroidi manualmente vedendo quale initializzazione mi dà il val. della f obj più basso

2) K-means e K-median non garantiscono di trovare un global optimum, è un algo greedy.

K-median ($\| \cdot \|_1$)

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

```

function [x,cluster,v] = kmedian(data,k,InitialCentroids)
l = size(data,1);

% initialize centroids
x = InitialCentroids;

% 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

% compute the objective function value
vold = 0;
for i = 1 : l
    vold = vold + norm(data(i,:)-x(cluster(i),:),1);
end

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

    % 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

    % update objective function
    v = 0;
    for i = 1 : l
        v = v + norm(data(i,:)-x(cluster(i),:),1);
    end

```

```

% stopping criterion
if vold - v < 1e-5
    break
else
    vold = v;
end
*
```

```

k = 3;
% multistart approach
vbest = inf;
xbest = [];
clusterbest = [];
maxiter = 100;
iter = 0;

while iter < maxiter
    InitialCentroids = 10*rand(k,2);
    [x,cluster,v] = kmedian(data,k,InitialCentroids);
    if v < vbest
        xbest = x;
        clusterbest = cluster;
        vbest = v;
    end
    iter = iter + 1;
end
*
```

Metodi per trovare la sol. negli unconstrained problems

Gradient Method (exact line search)

0) Scegli: x^0 (punto iniziale) e tolerance 10^{-6}

1) $x = x^0$, iter = 0

while True

Nel caso di un prob. quad con Q positive definite

$$a) v = f(x) \text{ e } g = \nabla f(x) \Rightarrow v = \frac{1}{2} \cdot x^T \cdot Q \cdot x + c \cdot x^T, g = Q \cdot x + c$$

b) se $\|g\| < \text{tolerance}$

b break

$$c) d = -g$$

Nel caso di un prob. quad con Q positive definite

$$\text{Trova la soluzione ottimale } t \text{ del problema } \min_{t>0} f(x + t \cdot d) \Rightarrow t = -\frac{g^T \cdot d}{d^T \cdot Q \cdot d}$$

$$x = x + t \cdot d, \text{ iter} = \text{iter} + 1$$

OSS:

- Quando il problema non è quadratrico la convergenza è costosa perché l'algoritmo cerca di scegliere al step size ottimale ad ogni iterazione (minimando la funzione).
- Due direzioni consecutive sono ortogonali 
- Se $f(x)$ quadratica e Q positive definite allora la convergenza è lineare.
- Se:
 - $f(x)$ coercive \rightarrow la sequenza verso x^* è bounded e $\nabla f(x^*) = 0$ (stationary point)
 - $f(x)$ coercive e convex \rightarrow la sequenza verso x^* è bounded e x^* glob. optimum
 - $f(x)$ strongly convex \rightarrow la sequenza converge verso x^* e x^* è un glob. optimum

Gradient Method (Armijo in exact line search)

0) Scegli: x^0 (punto iniziale) e $\alpha, \gamma \in (0,1)$ e $\bar{\epsilon} > 0$ e tolerance 10^{-6}

1) $x = x^0$, iter = 0

while True

a) $v = f(x^k)$ e $g = \nabla f(x^k)$

b) se $\|g\| < \text{tolerance}$

bneall

c) $d = -g$

$t = \bar{t}$

while $f(x + t \cdot d) > v + \alpha \cdot g^T \cdot t$

$t = \gamma \cdot t$

$x = x + t \cdot d$, $\text{iter} = \text{iter} + 1$

OSS:

- Lo step size non è preciso come nell'exact line search, ma viene approssimato
- Se $f(x)$ coercive \rightarrow la sequenza verso x^* è bounded e $\nabla f(x^*) = 0$

Conjugate Gradient Method (Quadratic Case)

1) Scegli: x^0 , e tolerance $+ \epsilon^{-6} (10^{-6})$

2) $x = x_0$ e $\text{iter} = 0$

while True

a) $v = \frac{1}{2} \cdot x^T \cdot Q \cdot x + c \cdot x^T$, $g = Q \cdot x + c$

b) se $\|g\| < \text{tolerance}$

bneall

c) Se $\text{iter} = 0 \Rightarrow d = -g$

altrimenti $\beta = \frac{\|g\|^2}{\|g_{\text{prev}}\|^2}$ e $d = -g + \beta \cdot d_{\text{prev}}$

d) $t = \frac{\|g\|^2}{d^T \cdot Q \cdot d}$

$$e) \quad x = x + t \cdot d, \quad iter = iter + 1, \quad d_pnew = d, \quad g_pnew = g$$

OSS:

- Trova il minimo globale in al più n iterazioni
- Se Q ha n autovalori, CG trova il minimo globale in al più n iterazioni
- Solitamente converge più velocemente del gradient method

Newton Method

o) Scegli $x^0 \in \mathbb{R}^n$ e tolerance $+e^{-6}$ ($+10^{-6}$)

1) $x = x^0$ e $iter = 0$

while True

a) $v = f(x)$, $g = \nabla f(x)$, $H = \nabla^2 f(x)$

b) se $\|g\| < \text{tolerance}$

break

c) $d = -\frac{\nabla f(x)}{\nabla^2 f(x)}$

d) $x = x + d$, $iter = iter + 1$

Drawbacks of Newton method:

- at each iteration we need to compute both the gradient $\nabla f(x^k)$ and the hessian matrix $\nabla^2 f(x^k)$
- local convergence: if x^0 is too far from the optimum x^* , then the generated sequence can be not convergent to x^*

Newton Method con line search (Annullo Inexact)

0) Imposta $\alpha, \gamma \in (0,1)$, $\bar{t} > 0$, scegli x^0 , e tolerance $\epsilon \cdot 10^{-6}$

1) $x = x^0$ e $\text{iter} = 0$

while true

a) $v = f(x)$, $g = \nabla f(x)$ e $H = \text{Hessian Matrix}(x) \approx \nabla^2 f(x)$

b) se $\|g\| < \text{tolerance}$

break

c) $d = -g / H$, $t = \bar{t}$

d) finche' $f(x + t \cdot d) > v + (\alpha \cdot g^T \cdot d \cdot t)$:

$$t = \gamma \cdot t$$

e) $x = x + t \cdot d$, $\text{iter} = \text{iter} + 1$

$$f(x, y) = y^4 + x^3 + 3x^2 + 4y^2 - 4xy - 5y + 8$$

$$\begin{aligned} g(x, y) &= \begin{bmatrix} 3x^2 + 6x - 4y \\ 4y^3 + 8y - 4x - 5 \end{bmatrix}, \\ H(x, y) &= \begin{bmatrix} 6x+6^* & -4 \\ -4 & 12y^2+8^* \end{bmatrix} \end{aligned}$$

- * Derivate basandosi su y invece che x
- * Derivate basandosi su x invece che y
- * Derivate "normali"

OSS: Se f strongly convex, la sequenza converge verso l'ottimo globale x^* e, inoltre, se $\alpha \in (0, 1/2)$ e $\bar{t} = 1$ allora la convergenza è quadratica!

Derivative Free Methods (Directional direct-search method)

Sono metodi che non usano le derivate (quindi while quando dobbiamo minimizzare una funzione non derivabile).

Ad ogni iterazione calcolano la funzione obiettivo su un numero fisso di punti cercando di capire la direzione per minimizzare la funzione.

0) Scegli: x^0 , step size $t_0 > 0$, $\beta \in (0, 1)$, tolerance $\epsilon > 0$ e una base positiva D , $k=0$

1) $x = x^0$, $t = t^0$, $v = f(x)$, $\text{iter} = 0$

while $t > \epsilon$:

A positive basis is a set of vectors $\{v^1, \dots, v^p\} \subset \mathbb{R}^n$ such that:

- any $x \in \mathbb{R}^n$ is a conic combination of v^1, \dots, v^p , i.e., there exist $\alpha_1, \dots, \alpha_p \geq 0$ such that $x = \sum_{i=1}^p \alpha_i v^i$
- for any $i = 1, \dots, p$, v^i is not a conic combination of others v^1, \dots, v^p .

a) $\text{new_v} = v$, $\text{iter} = \text{iter} + 1$
 n° colonne di D

b) for $i = 1 : \text{size}(D, 2)$

$\text{newx} = x + t \cdot D(:, i)$

$\text{newv} = f(\text{newx})$

se $\text{newv} < v$

$x = \text{newx}$, $v = \text{newv}$

break

c) se $\text{newv} \geq v$

$t = \beta \cdot t$

oss:

• Derivative free methods possono essere più lenti dei derivative-based methods

perché richiedono la valutazione della funzione obiettivo in più punti.

Inoltre, possono essere meno efficienti in high-dimensional problems essendo che

il numero di punti richiesti per la ricerca nello spazio aumenta esponenzialmente

con il numero di dimensioni.

Metodi per trovare la sol. nei constrained problems

Active-set Method

Questo metodo risolve ad ogni iterazione un problema quadratico con solo equality constraints

b) Scegli un punto iniziale $x^0 \in \mathbb{R}^n$, setta il working set W_0 dove $\{i : A_i x^0 = b_i\}$ e $k=0$

1) Trova la sol. ottimale y^k del problema

$$\begin{cases} \min \frac{1}{2} x^T Q x + c^T x \\ A_i x = b_i \quad \forall i \in W_k \end{cases}$$

2) Se $y^k \neq x^k$ (soluzione precedente o punto iniziale) \rightarrow vai step 2

altrimenti vai step 3

3) a) Se y^k è feasible, allora $t_k = 1$

altrimenti $t_k = \min \left\{ \frac{b_i - A_i x^k}{A_i(y^k - x^k)} : i \notin W_k, A_i(y^k - x^k) > 0 \right\},$

b) $x^{k+1} = x^k + t_k(y^k - x^k)$, $W_{k+1} = W_k \cup \{i \in W_k : A_i x^{k+1} = b_i\}$, $k=k+1$

c) Vai allo step 1

4) a) Calcola i kkt multipliers μ^k relativi a y^k \Rightarrow

$$\begin{cases} \nabla f(x) + \sum_{j=1}^p \mu_j \nabla h_j(x^*) = 0 \\ h(x) = 0 \end{cases}$$

b) Se $\mu^k \geq 0 \Rightarrow$ stop

altrimenti $x^{k+1} = x^k$, $k=k+1$, rimuovo da w il const. relativo al μ_i minore

Esempio

1) $\begin{cases} \min x_1^2 + x_2^2 + x_3^2 + x_2 x_3 - 5x_1 - 4x_2 - 3x_3 \\ x_2 = 1 \\ x_3 = 1 \end{cases}$

$$y^3 = x^3$$

$$\begin{cases} \begin{pmatrix} 2x_1 - 5 \\ 2x_2 + x_3 - 4 \\ 2x_3 + x_2 - 3 \end{pmatrix} + \mu_1 \begin{pmatrix} -\frac{5}{2} \\ 0 \\ 0 \end{pmatrix} + \mu_2 \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix} + \mu_3 \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} = 0 \\ x_1 = \frac{5}{2} \\ x_2 = 1 \\ x_3 = 1 \end{cases} \quad \begin{cases} \mu_1 = 0 \\ \mu_2 = -1 \\ \mu_3 = 0 \\ x_1 = \frac{5}{2} \\ x_2 = 1 \\ x_3 = 1 \end{cases}$$

Penalty Method

matrice e vettore dei vincoli

o) Define Q, c, A, b

Define T, ϵ_0 , tolerance (es: $10^{-6} - 10^{-6}$)

Define x_0

1) $\epsilon_{\text{ps}} = \epsilon_0, x = x_0, \text{iter} = 0$

while true

$x = \text{fminunc}(\text{G P-eps}, x, \text{options})$

verifica se

$x \in \Omega \leftarrow \text{infeas} = \max(Ax - b)$

se $\text{infeas} < \text{tolerance} \rightarrow \text{break}$

altrimenti $\rightarrow \epsilon_{\text{ps}} = T \cdot \epsilon_{\text{ps}}, \text{iter} += 1$

end

OSS:

$$\cdot P_\epsilon(x) = \begin{cases} f(x) & \text{if } x \in \Omega \\ > f(x) & \text{if } x \notin \Omega \end{cases}$$

• Se il problema è convex, anche (P_ϵ) è convex

• Se x_ϵ^* risolve (P_ϵ) e $x_\epsilon^* \in \Omega \Rightarrow x_\epsilon^*$ è ottimo anche per (P)

fun $[V, g] = p_eps(x)$

$$V = \frac{1}{2} x^T Q x + c^T x$$

$$g = Q \cdot x + c$$

for $i : \text{size}(A, 1)$

$$\quad V = V + \left(\frac{1}{\epsilon_{\text{ps}}}\right) \cdot \max(0, A(i, :) \cdot x - b(i))^2$$

$$\quad g = g + \left(\frac{2}{\epsilon_{\text{ps}}}\right) \cdot \max(0, A(i, :) \cdot x - b(i)) \cdot A(i, :)^T$$

corrisponde alle def. del problema

$$\begin{cases} \min f(x) + \frac{1}{\epsilon} \cdot \sum_{i=1}^m (\max(0, g_i(x)))^2 \\ x \in \mathbb{R}^n \end{cases}$$

ϵ constraints

dove $V = f_{\text{obj}}$ e g è la derivata

Logarithmic barrier Method

matrice e vettore dei vincoli

o) Define Q, c, A, b

Define T, ϵ_0 , tolerance (es: $10^{-6} - 10^{-6}$)

Define x_0 , $m = \text{size}(A, 1) \rightarrow$ n° righe in A

1) $\epsilon_{\text{ps}} = \epsilon_0, x = x_0, \text{iter} = 0$

while true

$x = \text{fminunc}(\text{G log bar}, x, \text{options})$

$\text{gap} = m \cdot \epsilon_{\text{ps}} = \text{optimality GAP}$

fun $[V, g] = \log \text{bar}(x)$

$$V = \frac{1}{2} x^T Q x + c^T x$$

$$g = Q \cdot x + c$$

for $i : \text{length}(b)$

$$\quad V = V - \epsilon_{\text{ps}} \cdot \log(b(i) - A(i, :) \cdot x)^2$$

$$\quad g = g + \left(\frac{\epsilon_{\text{ps}}}{b(i) - A(i, :) \cdot x}\right) \cdot A(i, :)^T$$

se gap < tolerance \rightarrow break

altrimenti \rightarrow $\text{eps} = \tau \cdot \text{eps}$, iter += 1
end

Oss:

$B(x)$ è la logarithmic barrier function ed è convex

corrisponde alla def. del problema

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

constraints

dove $v = f_{\text{obj}}$ e g è la derivata

Multiobj optimization

$$\left\{ \begin{array}{l} \min f(x) = (f_1(x), f_2(x), \dots, f_p(x)) \\ x \in \Omega \end{array} \right. \quad (P)$$

Minimum definitions

- 1) Ideal Minimum: corrisponde a una soluzione ottimale per tutte le obj. functions.
- 2) Minimum: corrisponde a una soluzione ottimale per almeno una obj. function
- 3) Weak minimum: Non corrisponde necessariamente al minimo di una funzione obiettivo, ma rappresenta un punto che indica che migliorando una qualsiasi funzione obiettivo, una delle altre funzioni obiettivo peggiora.

Oss: Ideal Minimum \Rightarrow Minimum \Rightarrow Weak Minimum

Esistenza di un minimo (Generalized Weinstass teo)

- 1) Se f_i continua $\forall i = 1 \dots p$ e Ω chiuso e limitato $\Rightarrow \exists$ minimum
- 2) Se f_i continua $\forall i = 1 \dots p$, Ω chiusa e f_j coercive per qualche $j \in \{1 \dots p\}$ $\Rightarrow \exists$ minimum

Optimality condition (minimum)

$x^* \in \Omega$ è un minimo per P \Leftrightarrow il problema

auxiliario seguente ha come valore ottimale 0

$$\left\{ \begin{array}{l} \max \sum_{i=1}^p \varepsilon_i = - \min \sum_{i=1}^p \varepsilon_i \\ f_i(x) + \varepsilon_i \leq f_i(x^*) \quad \forall i = 1, \dots, p \\ x \in \Omega \\ \varepsilon \geq 0 \end{array} \right. \rightarrow \text{constraints del prob. originale}$$

Per definizione di punto minima accetto solo $\varepsilon = 0$ perché:

- 1) $f_i(x) \leq f_i(x^*) \quad \forall i = 1 \dots p$
- 2) $\exists j \in \{1 \dots p\} \quad f_j(x) < f_j(x^*)$

Optimality condition (weak minimum)

$x^* \in \Omega$ è un weak minimum per P \Leftrightarrow il problema

auxiliario seguente ha come valore ottimale 0

$$\left\{ \begin{array}{l} \max v = - \min (-v) \\ v \leq \varepsilon_i \\ f_i(x) + \varepsilon_i \leq f_i(x^*) \quad \forall i = 1, \dots, p \\ x \in \Omega \\ \varepsilon \geq 0 \end{array} \right. \rightarrow \text{constraints del prob. originale}$$

```

%% data
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

% Get the dimensions of C and A
[p, n] = size(C);
m = size(A, 1);

% Set options for the optimization function
options = optimset('Display', 'off');

% 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)];
[~, 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)

```

Esempio :

Verifica se $x^* = (1, 4, 4)$ è un punto minima del problema:

$$\begin{cases} \min (x_1 + 2x_2 - 3x_3, -x_1 - x_2 - x_3, -4x_1 - 2x_2 + x_3) \\ x_1 + x_2 + x_3 \leq 10 \\ x_3 \leq 5 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

Handwritten derivation:

- min $\begin{bmatrix} 0 \\ -1 \\ -1 \end{bmatrix} \begin{bmatrix} x_1, x_2, x_3, \varepsilon_1, \varepsilon_2, \varepsilon_3 \end{bmatrix}$

$\begin{bmatrix} 1 & 2 & -3 \\ -1 & -1 & -1 \\ -4 & -2 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \leq \begin{bmatrix} 1 & 2 & -3 \\ -1 & -1 & -1 \\ -4 & -2 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 4 \\ 4 \end{bmatrix}$

$\downarrow f(x) \quad \downarrow \varepsilon \quad \downarrow x \quad \downarrow b$

$\begin{bmatrix} 4 & 4 & 1 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix} \leq \begin{bmatrix} 10 \\ 5 \\ 0 \\ 0 \end{bmatrix}$

\downarrow Constraints

$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

max $\sum_{i=1}^p \varepsilon_i$
 $f_i(x) + \varepsilon_i \leq f_i(x^*) \quad \forall i = 1, \dots, p$
 $x \in \Omega$
 $\varepsilon \geq 0$

First order optimality conditions

1) Unconstrained Problem

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

Necessarie:

Se x^* è un weak minimum di (P), esiste $\xi^* \in \mathbb{R}^p$ t.c.

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

Sufficienti:

Se P convex e (x^*, ξ^*) risolve (S), allora x^* è un weak minimum di P.

2) Constrained Problem

$$\begin{cases} \min f(x) = (f_1(x), f_2(x), \dots, f_p(x)) \\ g_j(x) \leq 0 \quad \forall j = 1, \dots, m \\ h_k(x) = 0 \quad \forall k = 1, \dots, q \end{cases}$$

Necessarie:

Se x^* è un weak minimum di P e ACQ holds in x^* , allora $\exists \xi^* \in \mathbb{R}^p$, $\lambda^* \in \mathbb{R}^m$ e $\mu^* \in \mathbb{R}^q$ t.c. $(x^*, \xi^*, \lambda^*, \mu^*)$ risolve il KKT system.

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

Sufficienti:

Se (P) convex e $(x^*, \xi^*, \lambda^*, \mu^*)$ risolve il KKT system, allora x^* è un weak min.

Scalarization Method

Definiamo un vettore di pesi $d \geq 0$ associati alle funzioni obiettivo, t.c.

Il problema scalarizzato è il seguente:

$$\begin{cases} \min \sum_{i=1}^p \alpha_i f_i(x) \quad (P_d) \\ x \in \Omega \end{cases}$$

$$\sum_{i=1}^p d_i = 1$$

Theorem

- $\bigcup_{\alpha \geq 0} S_\alpha \subseteq \{\text{weak minima of } (P)\}$
- $\bigcup_{\alpha > 0} S_\alpha \subseteq \{\text{minima of } (P)\}$

Lisolvere S_α ci fa trovare un

sottosistema di weak minima

2 minima, non tutti, però

abbiamo dei casi particolari:

Theorem

- If (P) is linear, then $\{\text{weak minima of } (P)\} = \bigcup_{\alpha \geq 0} S_\alpha$ and $\{\text{minima of } (P)\} = \bigcup_{\alpha > 0} S_\alpha$
- If (P) is convex, then $\{\text{weak minima of } (P)\} = \bigcup_{\alpha \geq 0} S_\alpha$
- If (P) is convex and f_i is strongly convex for any $i = 1, \dots, p$, then $\{\text{minima of } (P)\} = \{\text{weak minima of } (P)\} = \bigcup_{\alpha > 0} S_\alpha$

Esempio:

$$\left\{ \begin{array}{l} \min(x_1 - x_2, x_1 + x_2) \\ -2x_1 + x_2 \leq 0 \\ -x_1 - x_2 \leq 0 \\ 5x_1 - x_2 \leq 6 \end{array} \right. \rightarrow \left\{ \begin{array}{l} \min \alpha_1(x_1 - x_2) + \alpha_2(x_1 + x_2) \\ -2x_1 + x_2 \leq 0 \\ -x_1 - x_2 \leq 0 \\ 5x_1 - x_2 \leq 6 \\ \alpha_1 + \alpha_2 = 1 \end{array} \right. \rightarrow \text{Graph of the feasible region } S \quad \left. \begin{array}{l} \alpha_2 = 1 - \alpha_1 \\ \alpha_1 + \alpha_2 = 1 \end{array} \right.$$

$$\left\{ \begin{array}{l} \min \alpha_1(x_1 - x_2) + (1 - \alpha_1)(x_1 + x_2) \\ -2x_1 + x_2 \leq 0 \\ -x_1 - x_2 \leq 0 \\ 5x_1 - x_2 \leq 6 \\ \alpha_1 + \alpha_2 = 1 \end{array} \right. \quad \left\{ \begin{array}{l} \min x_1 + (1 - 2\alpha_1)x_2 \\ -2x_1 + x_2 \leq 0 \\ -x_1 - x_2 \leq 0 \\ 5x_1 - x_2 \leq 6 \\ \alpha_1 + \alpha_2 = 1 \end{array} \right.$$

Considero i casi:

$\left. \begin{array}{l} \text{weak minima} \\ \text{minima} \end{array} \right\}$	$\left. \begin{array}{l} \text{a)} \alpha_1 = 0 \rightarrow \alpha_2 = 1 \\ \text{b)} \alpha_1 = 1 \rightarrow \alpha_2 = 0 \\ \text{c)} 0 < \alpha_1 < 1 \rightarrow 0 < \alpha_2 < 1 \end{array} \right\}$	uso desmos considerando la funzione $x_1 + (1 - 2\alpha_1)x_2 = v$ impostando α e v parametrici
--	--	---

a) weak minima = segment $((0,0), (1, -1))$

Segment $((0,0), (1, -1))$

b) weak minima = segment $((0,0), (2, 4))$

Segment $((0,0), (2, 4))$

c) minima = i) $0 < \alpha_1 < \frac{3}{4} \Rightarrow S_\alpha = (0,0)$

Segment $((0,0), (1, 1))$

$$2) \alpha_1 = 1/4 \Rightarrow S\alpha = \text{segment}((0,0)(2,4)) \rightarrow \boxed{\text{segment}(w_1 w_2, (z_1 z_2))}$$

$$3) \frac{3}{4} < \alpha_1 < 1 \Rightarrow S\alpha = (2,4)$$

```

%% data
A = [ -2 1 ;
      -1 -1 ;
      5 -1 ];
b = [ 0 ; 0 ; 6 ];

%% plot the feasible region
plot([0 2], [0 4], '-k');
plot([2 1], [4 -1], '-k');
plot([1 0], [-1 0], '-k');

% Set options for the optimization function
options = optimset('Display', 'off');

%% Solve the scalarized problem with 0 < alfa < 1
for alfa = 0.001 : 0.001 : 0.999
    x = linprog([1; 1 - 2 * alfa], A, b, [], [], [], options);
    plot(x(1), x(2), 'g.');
end

%% solve the scalarized problem with alfa = 0
alfa = 0;
x0 = linprog([1; 1 - 2 * alfa], A, b, [], [], [], options);
plot(x0(1), x0(2), 'r*');

%% solve the scalarized problem with alfa = 1
alfa = 1;
x1 = linprog([1; 1 - 2 * alfa], A, b, [], [], [], options);
plot(x1(1), x1(2), 'bo');

```

Goal Method

Definiamo l'ideal point con z . Spesso i problemi non hanno un ideal point, quindi vogliamo trovare un punto (minima o weak minima) che sia il più vicino possibile a z :

$$\left\{ \begin{array}{l} \min_{x \in \Omega} \|f(x) - z\|_s \\ \end{array} \right. \quad \text{with } s \in [1, +\infty]. \quad (G)$$

teorema:

- 1) Se $s \in [+, +\infty)$ \Rightarrow la soluzione del problema G è un minima del problema (P)
- 2) Se $s = +\infty$ \Rightarrow la soluzione del problema G è un weak minima del problema (P)

Esempio con problema lineare $\left\{ \begin{array}{l} \min Cx \\ Ax \leq b \end{array} \right.$

$\| \cdot \|_2$) Il problema (G) diventa equivalente a un quadratic problem

$$\left\{ \begin{array}{l} \min \frac{1}{2} \|Cx - z\|_2^2 = \frac{1}{2} x^T C^T C x - x^T C^T z + \frac{1}{2} z^T z \\ Ax \leq b \end{array} \right.$$

$\| \cdot \|_1$) Il problema (G) diventa equivalente a un linear problem

$$\begin{cases} \min \sum_{i=1}^p y_i \\ y_i \geq C_i x - z_i \quad \forall i = 1, \dots, p \\ y_i \geq z_i - C_i x \quad \forall i = 1, \dots, p \\ Ax \leq b \end{cases}$$

$\| \cdot \|_\infty$) il problema diventa equivalente a un linear problem

$$\begin{cases} \min_{x,y} y \\ y_i \geq C_i x - z_i \quad \forall i = 1, \dots, p \\ y_i \geq z_i - C_i x \quad \forall i = 1, \dots, p \\ Ax \leq b \end{cases}$$

```
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 ] ;

% size of matrices C and A
p = size(C,1);
n = size(C,2);
m = size(A,1);

% create options for linprog function
options = optimset('Display','off');

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

%% goal method

% 1-norm
gml = linprog([zeros(n,1);ones(p,1)], [C -eye(p); -C -eye(p); A zeros(m,p)], [z;-z;b], ...
[],[],[],[],[],options);
gml = gml(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)
```

Non Cooperative Game theory

$$\text{Player 1: } \min_{x \in X} f_1(x, y)$$

$$\text{Player 2: } \min_{y \in Y} f_2(x, y)$$

↑
cost function
strategy of P₁
y ∈ Y → strategy of P₂
↓ strategies set of strategies

Nash Equilibrium

Coppia di strategie (\bar{x}, \bar{y}) t.c.

- 1) \bar{x} è la migliore risposta del player 1 alla strategia \bar{y} del player 2 $f_1(\bar{x}, \bar{y}) = \min_{x \in X} f_1(x, \bar{y})$
- 2) \bar{y} è la migliore risposta del player 2 alla strategia \bar{x} del player 1 $f_2(\bar{x}, \bar{y}) = \min_{y \in Y} f_2(\bar{x}, y)$

Matrix Games

È un two-person non-cooperative game dove:

- 1) X e Y sono insiemi finiti di strategie $X = \{1, \dots, m\}, Y = \{1, \dots, n\}$
- 2) $f_2 = -f_1$ (zero-sum game)

Il gioco può essere rappresentato tramite una matrice C di dimensione $m \times n$, dove $c_{i,j}$ è la quantità di soldi che P_1 paga a P_2 se P_1 sceglie la strategia i e P_2 sceglie la strategia j .

		Player 2			
		1	2	3	
Player 1	1	1	-1	0	
	2	3	-2	-1	
	3	2	3	-2	

Nash eq. ↓

		Player 2		
		1	2	
Player 1	1	1	1	
	2	1	-1	

Uma strategia che porta ad entrambi i player un vantaggio.
Ho eliminato le Strictly Dominated Strategies!

- 1) Rimuovo le righe che hanno costo maggiore rispetto un'altra (perché costano di più al player 1 quindi non le giocherebbe mai)
- 2) Rimuovo le colonne che hanno costo minore rispetto un'altra (perché portano meno profitto al player 2, quindi non le giocherebbe mai)

Oss: Ho una sola matrice perché il gioco è uno zero-sum-game!

		Player 2			
		1 (odd)	(even)	2	
Player 1	1 (odd)	1	-1	1	
	2 (even)	1	-1	1	

→ Non ci sono né Strictly Dominated Strategies, né Nash equilibria!
Se P_1 sceglie una strategia è sconveniente per P_2 e viceversa.

Mixed Strategy

Invece di considerare le strategie come "Sì" o "No", consideriamo la loro probabilità

$$X = \{x \in \mathbb{R}^m : x \geq 0, \sum_{i=1}^m x_i = 1\} \quad \text{e} \quad Y = \{y \in \mathbb{R}^n : y \geq 0, \sum_{j=1}^n y_j = 1\}$$

Oss: Se una strategia $x_i = 1$ e le altre hanno valore 0 ($\forall k \neq i, x_k = 0$), allora x_i è detta pure strategy.

le cost function, considerando le mixed strategies, cambiano nel seguente modo:

$$1) f_1(x, y) = x^T C y \rightarrow \text{cioè sto considerando } \sum_{i=1}^m \sum_{j=1}^n x_i c_{ij} y_j$$

$$2) f_2(x, y) = -x^T C y$$

e si chiamano "expected cost".

Mixed Nash Equilibrium

$(\bar{x}, \bar{y}) \in X \times Y$ è una mixed strategies Nash equilibrium sse:

$$\begin{cases} \bar{x} \text{ is an optimal solution of } \min_{x \in X} \max_{y \in Y} x^T C y \\ \bar{y} \text{ is an optimal solution of } \max_{y \in Y} \min_{x \in X} x^T C y \end{cases}$$

II

1. The problem $\min_{x \in X} \max_{y \in Y} x^T C y$ is equivalent to the linear programming problem

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

2. The problem $\max_{y \in Y} \min_{x \in X} x^T C y$ is equivalent to the linear programming problem

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

OSS:

- P_2 è il duale di P_1
- Ogni Matrix Game ha almeno una Mixed Nash Equilibrium!
- ↳ Perché consiste nella risoluzione di 2 problemi lineari, quindi esiste sempre una soluzione.

% cost matrix

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

% solve the LP problem

→ % Get the dimensions of the cost matrix
 $[m, n] = \text{size}(C);$

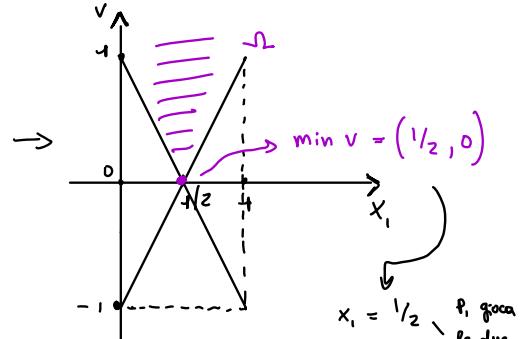
% Solve the linear programming problem using the linprog function
 $[\text{sol}, v, \sim, \sim, \lambda] = \text{linprog}([\text{zeros}(m, 1); 1], \dots$
 $[C, -\text{ones}(n, 1)], \text{zeros}(n, 1), \dots$
 $[\text{ones}(1, m), 0], 1, \dots$
 $[\text{zeros}(m, 1); -\text{inf}], []);$

% Extract the Nash equilibrium values
 $x = \text{sol}(1:m)$
 $y = \lambda$

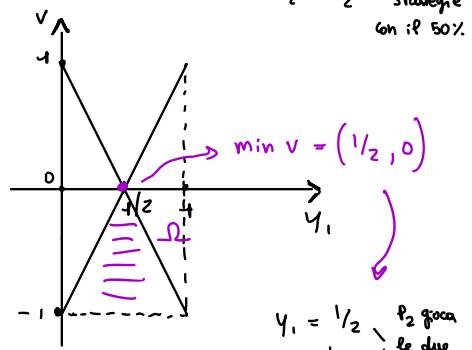
Esempio:

		p_2
		1 (odd) 2 (even)
p_1	1 (odd)	1 -1
	2 (even)	-1 1

$$\left\{ \begin{array}{l} \min V \\ V \geq x_1 - x_2 \\ V \geq -x_1 + x_2 \\ x_1 \geq 0 \\ x_2 \geq 0 \\ x_1 + x_2 = 1 \Rightarrow x_1 = 1 - x_2 \end{array} \right. \rightarrow \left\{ \begin{array}{l} \min V \\ V \geq 2x_1 - 1 \\ V \geq 1 - 2x_1 \\ 0 \leq x_1 \leq 1 \end{array} \right.$$



$$\left\{ \begin{array}{l} \max W \\ W \leq y_1 - y_2 \\ W \leq -y_1 + y_2 \\ y_1 \geq 0 \\ y_2 \geq 0 \\ y_1 = 1 - y_2 \end{array} \right. \rightarrow \left\{ \begin{array}{l} \max W \\ W \leq 2y_1 - 1 \\ W \leq 1 - 2y_1 \\ 0 \leq y_1 \leq 1 \end{array} \right.$$



Entrambi: problemi hanno una sola soluzione \Rightarrow 1 mixed Nash equilibria!

Bimatrix Games

È un two-person noncooperative game dove:

- 1) l'insieme delle pure strategies è finito, e l'insieme delle mixed strategies sono:
 - a) $X = \{x \in \mathbb{R}^m : x \geq 0, \sum_{i=1}^m x_i = 1\}$ le cost matrix sono diverse
 - b) $Y = \{y \in \mathbb{R}^n : y \geq 0, \sum_{j=1}^n y_j = 1\}$
- 2) $f_2 \neq f_1$, (non-zero sum game) $\rightarrow f_1(x, y) = x^T c_1 y$ e $f_2(x, y) = x^T c_2 y$
- 3) c_1 e c_2 sono due cost matrix $m \times n$.

Oss: Ogni bimatrix game ha almeno uno mixed strategies Nash equilibrium.

Le **strictly dominated strategies** nel caso dei bimatrix games sono:

- 1) le righe con valori maggiori in C_1
- 2) le colonne con valori maggiori in C_2

Dobbiamo eliminarle

$$C_1 = \begin{pmatrix} 5 & 1 \\ 10 & 2 \end{pmatrix} \quad C_2 = \begin{pmatrix} 5 & 10 \\ 1 & 2 \end{pmatrix}$$

Nash Equilibrium Example

$$C_1 = \begin{pmatrix} -5 & 0 \\ 0 & -1 \end{pmatrix} \quad C_2 = \begin{pmatrix} -1 & 0 \\ 0 & -5 \end{pmatrix}$$

- 1) Ho **due pure strategies**:
nash equilibria

a) $x = (1, 0)$ $y = (1, 0)$ → la strategia 1 è la migliore per 1

b) $x = (0, 1)$ $y = (0, 1)$ → la strategia 2 è la migliore per 2

entrambi

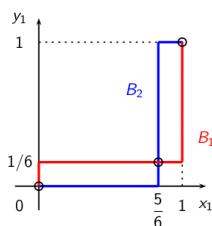
entrambi

- 2) Ho una **mixed strategy** (La ho usato il **Best Response Mapping**)

nash equilibria

$$\text{a) dato } y \text{ fissato} \rightarrow \left\{ \begin{array}{l} \min_x x^T C_1 y \\ x \in X \end{array} \right. \rightarrow \left\{ \begin{array}{l} \min_x -5x_1 y_1 - x_2 y_2 \\ x \in X \end{array} \right. \rightarrow \left\{ \begin{array}{l} \min_x (1 - 6y_1)x_1 + y_1 - 1 \\ 0 \leq x_1 \leq 1 \end{array} \right. \rightarrow B_1(y_1) = \begin{cases} 0 & \text{se } y_1 \in [0, 1/6] \\ [0, 1] & \text{se } y_1 = 1/6 \\ 1 & \text{se } y_1 \in (1/6, 1) \end{cases}$$

$$\text{b) dato } x \text{ fissato} \rightarrow \left\{ \begin{array}{l} \min_y x^T C_2 y \\ y \in Y \end{array} \right. \rightarrow \left\{ \begin{array}{l} \min_y -x_1 y_1 - 5x_2 y_2 \\ y \in Y \end{array} \right. \rightarrow \left\{ \begin{array}{l} \min_y (5 - 6x_1)y_1 + 5x_1 - 5 \\ 0 \leq y_1 \leq 1 \end{array} \right. \rightarrow B_2(x_1) = \begin{cases} 0 & \text{se } x_1 \in [0, 5/6] \\ [0, 1] & \text{se } x_1 = 5/6 \\ 1 & \text{se } x_1 \in (5/6, 1) \end{cases}$$



There are 3 Nash equilibria:

- $\bar{x} = (0, 1), \bar{y} = (0, 1)$ (pure strategies)
- $\bar{x} = (5/6, 1/6), \bar{y} = (1/6, 5/6)$ (mixed strategies)
- $\bar{x} = (1, 0), \bar{y} = (1, 0)$ (pure strategies)

Best Response Mappings

$$B_1 : Y \rightarrow X \quad e \quad B_2 : X \rightarrow Y$$

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

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

OSS: (\bar{x}, \bar{y}) è un nash equilibrium $\Leftrightarrow \bar{x} \in B_1(\bar{y})$ e $\bar{y} \in B_2(\bar{x})$

Le Nash equilibria sono date dall'intersezione dei grafici dei best response mappings.

KKT conditions per il Nash Equilibrium (usato per trovare il nash eq. anche per problemi complessi)
 (\bar{x}, \bar{y}) è un Nash Equilibrium sse $\exists \mu_1, \mu_2 \in \mathbb{R}$ t.c. :

$$\left\{ \begin{array}{l} C_1 \bar{y} + \mu_1 e \geq 0 \\ \bar{x} \geq 0, \quad \sum_{i=1}^m \bar{x}_i = 1 \\ \bar{x}_i(C_1 \bar{y} + \mu_1 e)_i = 0 \quad \forall i = 1, \dots, m \end{array} \right\} P_1$$

$$\left\{ \begin{array}{l} C_2^T \bar{x} + \mu_2 e \geq 0 \\ \bar{y} \geq 0, \quad \sum_{j=1}^n \bar{y}_j = 1 \\ \bar{y}_j(C_2^T \bar{x} + \mu_2 e)_j = 0 \quad \forall j = 1, \dots, n \end{array} \right\} P_2$$

dove $e = (-1 \dots -1)^T$

Caratterizzazione del Nash equilibria

Assumendo che $C_1 \succeq 0$ e $C_2 \succeq 0$ (posso ottenere questa condizione sottraendo da ogni valone delle matrici lo stesso numero) \rightarrow es: $(1, 1) \rightarrow (1-3, 1-3) \rightarrow (-2, -2)$

Se (\bar{x}, \bar{y}) è un Nash equilibrium, allora $\exists u > 0, v > 0$ t.c. $\tilde{x} = \bar{x}/u$ e $\tilde{y} = \bar{y}/v$

Risolvono il seguente sistema, che è equivalente al KKT system precedente.

$$\left\{ \begin{array}{l} \tilde{x} \geq 0, \quad C_1 \tilde{y} + e \geq 0, \quad \tilde{x}_i(C_1 \tilde{y} + e)_i = 0 \quad \forall i = 1, \dots, m \\ \tilde{y} \geq 0, \quad C_2^T \tilde{x} + e \geq 0, \quad \tilde{y}_j(C_2^T \tilde{x} + e)_j = 0 \quad \forall j = 1, \dots, n \end{array} \right. \quad (S)$$

* Complementary conditions
che "uniscono" x e y .

Se (\tilde{x}, \tilde{y}) risolvono (S) , allora $\left(\frac{\tilde{x}}{\sum_{i=1}^m \tilde{x}_i}, \frac{\tilde{y}}{\sum_{j=1}^n \tilde{y}_j} \right)$ è un nash equilibrio.

Questa versione del sistema è semplificata poiché non comprende le variabili μ ma solo x, y .

Inoltre, da questo sistema possiamo plottare due poliedri considerando solo le eq. e diseq. che coinvolgono x o y .

* $P = \left\{ (x_1, \dots, x_m) : \begin{array}{ll} x_i \geq 0 & \forall i = 1, \dots, m \\ (C_2^T x + e)_j \geq 0 & \forall j = m+1, \dots, m+n \end{array} \right\}$ Polyhedron of P_1

* $Q = \left\{ (y_{m+1}, \dots, y_{m+n}) : \begin{array}{ll} (C_1 y + e)_i \geq 0 & \forall i = 1, \dots, m \\ y_j \geq 0 & \forall j = m+1, \dots, m+n \end{array} \right\}$ Polyhedron of P_2

Da queste definizioni ottieniamo:

+ (\tilde{x}, \tilde{y}) risolve (S) sse $\tilde{x} \in P$ e $\tilde{y} \in Q$ e $\underline{\forall k = 1, \dots, m+n}$ ie k -th vincolo di P è

attivo in \tilde{x} oppure il k -th vincolo di Q è attivo in \tilde{y} .

Questo rappresenta le complementary conditions*

2) Se i vertici di P e Q sono non-degenerate, cioè quando ho esattamente m vincoli attivi per i vertici di P e n vincoli attivi per i vertici di Q , e (\tilde{x}, \tilde{y}) risolve il sistema (s) , allora \tilde{x} è un vertice di P e \tilde{y} è un vertice di Q .

Da questa definizione ottieniamo che se i vertici sono non-degenerate, allora ho un insieme finito di soluzioni per (s) che sono Nash equilibria e corrispondono ai vertici di P e Q .

Soluzione: Quindi, se $c_1 < 0$ e $c_2 < 0$, e i vertici di P e Q sono non-degenerate, possiamo trovare tutti i Nash Equilibria analizzando le coppie (x, y) dei vertici di P e Q , verificando se $\forall k = 1, \dots, m+n$ il k -th vincolo di P è attivo in \tilde{x} oppure il k -th vincolo di Q è attivo in \tilde{y} .

Esempio:

Siano $c_1 = \begin{pmatrix} -5 & 0 \\ 0 & -1 \end{pmatrix}$ $c_2 = \begin{pmatrix} -1 & 0 \\ 0 & -5 \end{pmatrix}$, sottraggo -1 a entrambe $c_1 = \begin{pmatrix} -6 & -1 \\ -1 & -2 \end{pmatrix}$ $c_2 = \begin{pmatrix} -2 & -1 \\ -1 & -6 \end{pmatrix}$ ottenendo $c_1, c_2 < 0$.

P e Q sono definiti come segue:

$$P = \{(x_1, x_2) : x_1 \geq 0, x_2 \geq 0, -2x_1 - x_2 + 1 \geq 0, -x_1 - 6x_2 + 1 \geq 0\}$$

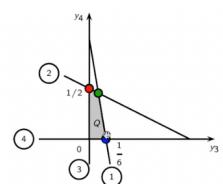
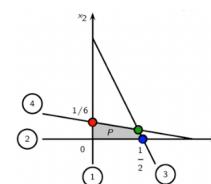
$$Q = \{(y_3, y_4) : -6y_3 - y_4 + 1 \geq 0, -y_3 - 2y_4 + 1 \geq 0, y_3 \geq 0, y_4 \geq 0\}$$

Vertici di P :

$$\boxed{(0, 1/6)}, \quad \boxed{(5/11, 1/11)}, \quad \boxed{(1/2, 0)}$$

Vertici di Q :

$$\boxed{(0, 1/2)}, \quad \boxed{(1/11, 5/11)}, \quad \boxed{(1/6, 0)}$$



Consideriamo i vertici $\neq (0,0)$ perché dobbiamo normalizzarne.

Per trovare i Nash equilibria devo normalizzarne $\left(\frac{\tilde{x}}{\sum_{i=1}^m \tilde{x}_i}, \frac{\tilde{y}}{\sum_{j=1}^n \tilde{y}_j} \right)$

Nash equilibria:

- 1) a) $(0, \frac{1}{6}) \rightarrow \left(\frac{0}{0+\frac{1}{6}}, \frac{\frac{1}{6}}{0+\frac{1}{6}} \right) \rightarrow (0,1) \Rightarrow \bar{x} = (0,1)$ 1° nash equilibria
 b) $(0, \frac{1}{2}) \rightarrow \left(\frac{0}{0+\frac{1}{2}}, \frac{\frac{1}{2}}{0+\frac{1}{2}} \right) \rightarrow (0,1) \Rightarrow \bar{q} = (0,1)$

- 2) a) $(\frac{5}{11}, \frac{1}{11}) \rightarrow \left(\frac{\frac{5}{11}}{\frac{5}{11}+1_{11}}, \frac{\frac{1}{11}}{\frac{5}{11}+1_{11}} \right) \rightarrow (\frac{5}{6}, \frac{1}{6}) \Rightarrow \bar{x} = (\frac{5}{6}, \frac{1}{6})$ 2° 11
 b) $(\frac{1}{11}, \frac{5}{11}) \rightarrow \left(\frac{\frac{1}{11}}{\frac{5}{11}+1_{11}}, \frac{\frac{5}{11}}{\frac{5}{11}+1_{11}} \right) \rightarrow (\frac{1}{6}, \frac{5}{6}) \Rightarrow \bar{q} = (\frac{1}{6}, \frac{5}{6})$

- 3) a) $(\frac{1}{2}, 0) \rightarrow \left(\frac{\frac{1}{2}}{\frac{1}{2}+0}, \frac{0}{\frac{1}{2}+0} \right) \rightarrow (1,0) \Rightarrow \bar{x} = (1,0)$ 3° 11
 b) $(\frac{1}{6}, 0) \rightarrow \left(\frac{\frac{1}{6}}{\frac{1}{6}+0}, \frac{0}{\frac{1}{6}+0} \right) \rightarrow (1,0) \Rightarrow \bar{q} = (1,0)$

Lemke - Howson algorithm

Quando $C_1 < 0$, $C_2 < 0$ e i vertici di P e Q sono non-degenerati, un Nash eq. può essere trovato usando il Lemke - Howson algo.

1) Setta $x=0$, $y=0$. Definisci: $I_x = \{1, \dots, m\}$, $I_y = \{m+1, \dots, m+n\}$.
 Scegli un index $h \in \{1, \dots, m\}$ set of P active const. set of Q active const.

2) Nel poliedro P , muovi h da x lungo l'arco dato dai constraint $I_x \setminus \{h\}$ verso il vertice adiacente x' .

3) Trova l'index k corrispondente al nuovo vincolo attivo in x' .
 Imposta $x = x'$ e $I_x = (I_x \setminus \{h\}) \cup \{k\}$

4) Se $h \notin I_y \Rightarrow$ Stop $\Rightarrow (x, y)$ risolve (s) \Rightarrow è un nash eq.

Altrimenti: Imposta $h = k$ e nel poliedro Q muovi h da y lungo l'arco dato dai vincoli $I_y \setminus \{h\}$ verso il vertice adiacente y'

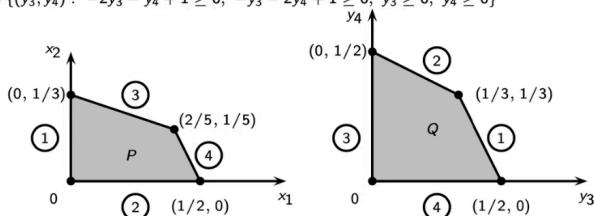
5) Trova l'index k corrispondente al nuovo vincolo attivo y'

Imposta $y = y'$ e $I_y = (I_y \setminus \{h\}) \cup \{k\}$

6) Se $k \notin I_x \Rightarrow$ stop $\Rightarrow (x, y)$ risolve (s) \Rightarrow trova un Nash eq.

Altimenti: imposta $h = k$ e vai allo step 2

Example. Consider the following bimatrix game: $C_1 = \begin{pmatrix} -2 & -1 \\ -1 & -2 \end{pmatrix}$ $C_2 = \begin{pmatrix} -1 & -2 \\ -3 & -1 \end{pmatrix}$
 $P = \{(x_1, x_2) : x_1 \geq 0, x_2 \geq 0, -x_1 - 3x_2 + 1 \geq 0, -2x_1 - x_2 + 1 \geq 0\}$
 $Q = \{(y_3, y_4) : -2y_3 - y_4 + 1 \geq 0, -y_3 - 2y_4 + 1 \geq 0, y_3 \geq 0, y_4 \geq 0\}$



Iter	h	k	x	I_x	y	I_y
1			(0, 0)	{1, 2}	(0, 0)	{3, 4}
2	1	4	(1/2, 0)	{2, 4}	(0, 0)	{3, 4}
3	4	2	(1/2, 0)	{2, 4}	(0, 1/2)	{2, 3}
4	2	3	(2/5, 1/5)	{3, 4}	(0, 1/2)	{2, 3}
5	3	1	(2/5, 1/5)	{3, 4}	(1/3, 1/3)	{1, 2}

Nash equil. $(2/3, 1/3)$ $(1/2, 1/2)$

Convex Games

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

↳ vincolo

L'insieme delle strategie per P_1 e P_2 è infinito, ma abbiamo un insieme di vincoli per esse.

Il gioco è detto convex se il problema di ottimizzazione per player è convex.

Teorema

- Se:
- 1) Ω_{P_1} e Ω_{P_2} sono chiuse, convesse e limitate
 - 2) la cost function $f_1(\cdot, y)$ è quasi-convex $\forall y \in \Omega_{P_2}$
 - 3) la cost function $f_2(x, \cdot)$ è quasiconvessa $\forall x \in \Omega_{P_1}$

Allora: Esiste almeno un Nash Equilibrium.

OSS: La quasi-convessità propriamente è necessaria.

KKT conditions per il Nash Equilibrium

Se (\bar{x}, \bar{y}) è un Nash equilibrium e ACQ holds sia in \bar{x} che in \bar{y} , allora $\exists \lambda^1, \lambda^2$

t.c.:

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

Se $(\bar{x}, \bar{y}, \lambda^1, \lambda^2)$ risolvono il KKT system e il gioco è convex, allora (\bar{x}, \bar{y}) è un Nash Equilibrium.

Menil Functions

Le Menil Functions sono funzioni che permettono di riformulare il Nash Equilibrium problem in un problema di ottimizzazione equivalente (generalmente non convex!).

Good News	Bad News
-----------	----------

Definiamo la Nikaido - Isoda function:

$$f(x, y, u, v) = \underbrace{f_1(u, y)}_{\text{Vantaggio di } P_1 \text{ nel cambiare}} - \underbrace{f_1(x, y)}_{\text{la strategia corrente (x) nella}} + \underbrace{f_2(x, v)}_{\text{Vantaggio di } P_2 \text{ nel cambiare}} - \underbrace{f_2(x, y)}_{\text{la strategia corrente (y) nella}}, \quad \text{dove } x, u \in \mathbb{R}^m \text{ e } y, v \in \mathbb{R}^n$$

Vantaggio di P_1 nel cambiare la strategia corrente (x) nella strategia (u) assumendo che P_2 gioca la strategia (y)

Vantaggio di P_2 nel cambiare la strategia corrente (y) nella strategia (v) assumendo che P_1 gioca la strategia (x)

Dalla Nikaido - Isoda function possiamo definire tre Menil functions:

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

$$\left\{ \begin{array}{l} \min \psi(x, y) \\ (x, y) \in X \times Y \end{array} \right. \rightarrow$$

- 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 $\psi(\bar{x}, \bar{y}) = 0$

```

function v = gap(z)
    % gap function of a bimatrix game
    global C1 C2
    % Get the dimensions of C1
    [m, n] = size(C1);
    % Extract x and y from the input vector
    x = z(1:m);
    y = z(m + 1:m + n);
    % Solution using linprog
    options = optimset('Display', 'off');
    [~, v1] = linprog(C1 * y, [], [], ones(1, m), 1, zeros(m, 1), [], options);
    [~, v2] = linprog(C2' * x, [], [], ones(1, n), 1, zeros(n, 1), [], options);
    v = x' * (C1 + C2) * y - v1 - v2;
    % Alternative solution without linprog
    % v = x' * (C1 + C2) * y - min(C1 * y) - min(C2' * x);
end

```

2) Regularized Gap Function ($\alpha > 0$) $\rightarrow \psi_\alpha(x, y) = \max_{u \in X, v \in Y} \left[-f(x, y, u, v) - \frac{\alpha}{2} \| (x, y) - (u, v) \|^2 \right].$

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

- ▶ 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 $\psi_\alpha(\bar{x}, \bar{y}) = 0$.

```

function v = reggap(z, alfa) % regularized gap function of a bimatrix game

global C1 C2;
% Get the dimensions of C1
[m, n] = size(C1);
% Extract x and y from the input vector
x = z(1:m);
y = z(m + 1:m + n);
% Set options for the optimization function
options = optimset('Display', 'off');

% Find the optimal value for x using quadprog
[~, v1] = quadprog(alfa * eye(m), C1 * y - alfa * x, [], [], ones(1, m), 1, ...
    zeros(m, 1), ones(m, 1), [], options);

% Find the optimal value for y using quadprog
[~, v2] = quadprog(alfa * eye(n), C2' * x - alfa * y, [], [], ones(1, n), 1, ...
    zeros(n, 1), ones(n, 1), [], options);

% Calculate the value of the regularized gap function
v = x' * (C1 + C2) * y - 0.5 * alfa * (norm(x)^2 + norm(y)^2) - v1 - v2;
end

```

3) D-Gap function ($B > \alpha > 0$) $\rightarrow \psi_{\alpha, \beta}(x, y) = \psi_\alpha(x, y) - \psi_\beta(x, y).$

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

- ▶ $\psi_{\alpha, \beta}$ 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 $\psi_{\alpha, \beta}(\bar{x}, \bar{y}) = 0$.

```

function v = Dgap(z, alfa, beta)
    % Calculate the difference between the regularized gap functions at alfa and beta
    v = reggap(z, alfa) - reggap(z, beta);
end

```