Linear SVM - primal model (Model)

```
close all; clear; clc;
A=[ 0.4952    6.8088  ];  //coppia, vado a capo, coppia, ecc
B=[ ];
nA = size(A,1);
nB = size(B,1);
T = [A ; B]; % training points
Q = [ 1 0 0 ;
    0 1 0 ;
    0 0 0 ];
D = [-A -ones(nA,1);
    B ones(nB,1) ] ;
d = -ones(nA+nB,1) ;
sol = quadprog(Q,zeros(3,1),D,d); % solve the problem
w = sol(1:2)
b = sol(3)
% plot the solution
xx = 0:0.1:10 ;
uu = (-w(1)/w(2)).*xx - b/w(2);
vv = (-w(1)/w(2)).*xx + (1-b)/w(2);
vvv = (-w(1)/w(2)).*xx + (-1-b)/w(2);
plot(A(:,1),A(:,2),'bo',B(:,1),B(:,2),'ro',xx,uu,'k-',xx,vv,'b-',xx,vvv,'r-','Linewidth',1.5)
axis([0 10 0 10])
```

Linear SVM - dual model (Model)

```
A=[ 0.4952    6.8088];
B=[     ];
nA = size(A,1);
nB = size(B,1);
T = [A ; B];  % training points
% define the problem
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
la = quadprog(Q,-ones(l,1),[],[],y',0,zeros(l,1),[]);   % solve the problem
% compute vector w
wD = zeros(2,1);
for i = 1 : l
  wD = wD + la(i)*y(i)*T(i,:)';
end
wD
% compute scalar b
ind = find(la > 1e-3) ;
i = 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);
```

```
figure
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 (dual model)')
```

Linear SVM - dual model (soft margin) (Model)

```
A=[2.650 8.95];
B=[7.7030  5.0965];
nA = size(A,1);
nB = size(B,1);
T = [A ; B];
% 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
la = quadprog(Q,-ones(l,1),[],[],y',0,zeros(l,1),C*ones(l,1),[]); % solve the problem
wD = zeros(2,1);
for i = 1 : l
  wD = wD + la(i)*y(i)*T(i,:)'; % compute vector w
end
indpos = find(la > 10^(-3));
ind = find(la(indpos) < C - 10^(-3));
i = indpos(ind(1));
bD = 1/y(i) - wD'*T(i,:)'; % compute scalar b
%% 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),'r*',...
    xx,uuD,'k-',xx,vvD,'b-',xx,vvvD,'r-','Linewidth',1)
axis([0 10 0 10])
title('Optimal separating hyperplane with soft margin')
% Compute the support vectors
supp = find(la > 10^(-3));
suppA = supp(supp <= nA);
suppB = supp(supp > nA);
% Compute the errors xi
for i=1:nA+nB
   if la(i) >0.001
      xi(i)= 1 - y(i)*(T(i,:)*wD +bD);
   else xi(i)=0;
   end
end
```

Nonlinear SVM (gausiian kernel) (Model)

```
A=[ ]; B=[ ];
nA = size(A,1);
nB = size(B,1);
T = [A ; B];
y = [ones(nA,1) ; -ones(nB,1)]; % labels
l = length(y);
C = 1 ; gamma = 1 ; K = zeros(l,l); % parameter
for i = 1 : l
   for j = 1 : l
      K(i,j) = exp(-gamma*norm(T(i,:)-T(j,:))^2);
   end
end
Q = zeros(l,l); % define the problem
for i = 1 : l
   for j = 1 : l
      Q(i,j) = y(i)*y(j)*K(i,j) ;
   end
end
[la,ov] = quadprog(Q,-ones(l,1),[],[],y',0,zeros(l,1),C*ones(l,1)); % solve the problem
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); % compute b
end
for xx = -2 : 0.01 : 2 %% plot the surface f(x)=0
   for yy = -2 : 0.01 : 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 (abs(s)< 10^(-2))
         plot(xx,yy,'g.');
      hold on
       end
   end
end
plot(A(:,1),A(:,2),'bo',B(:,1),B(:,2),'ro','Linewidth',5)
```

General Regression norm1 norm2 norm∞ (Model)

```
close all; clear; clc; //codice per pulire
%% data
data = [-5.0000  -96.2607 ];
x = data(:,1) ;
y = data(:,2) ;
l = length(x) ;
n = 4 ; % number of coefficients of polynomial
A = [ ones(l,1) x x.^2 x.^3 ]; % Vandermonde matrix
% 2-norm problem
z2 = inv(A'*A)*(A'*y) %la norm due è risolta solo con queste due righe di codice
p2 = A*z2; % regression values at the data
```

```
%% 1-norm problem. define the problem
c = [ zeros(n,1); ones(l,1) ];
D = [ A -eye(l); -A -eye(l) ];
d = [ y; -y ];
sol1 = linprog(c,D,d) ; % solve the problem
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) ];
solinf = linprog(c,D,d) ; % solve the problem
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');
```

Linear e-SV regression (solo e no slack variables) primal problem (Model)

```
data = [ ....];
x = data(:,1) ; y = data(:,2) ;
l = length(x) ; % number of points
epsilon = 0.5 ; %questo è quello che si modifica
% define the problem
Q = [ 1 0
     0 0 ];
c = [0;0];
D = [-x -ones(l,1)
     x  ones(l,1)];
d = epsilon*ones(2*l,1) + [-y;y];
sol = quadprog(Q,c,D,d); % solve the problem
w = sol(1); % compute w
b = sol(2); % compute b
z = w.*x + b ; % find regression and epsilon-tube
zp = w.*x + b + epsilon ;
zm = w.*x + b - epsilon ;
plot(x,y,'b.',x,z,'k-',x,zp,'r-',x,zm,'r-'); %% plot the solution
legend('Data','regression','\epsilon-tube',...
   'Location','NorthWest')
```

Linear e-SV regression - primal problem with slack variables (Model)

```
close all; clear; clc;
data = [];
x = data(:,1) ;
y = data(:,2) ;
l = length(x) ; % number of points
epsilon = 0.2 ; C = 10 ;  %%elementi da modificare
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];
sol = quadprog(Q,c,D,d,[],[],[-inf;-inf;zeros(2*l,1)],[]); % solve the problem
```

Linear e-SV regression - primal problem with slack variables

```
w = sol(1); % compute w
b = sol(2); % compute b
% compute slack 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')
```

Linear e-SV regression – Dual problem with slack variables (Model)

```
close all; clear; clc;
data = [....];
x = data(:,1) ;
y = data(:,2) ;
l = length(x) ; % number of points
epsilon = 0.2 ; C = 10; % parameters da cambiare
% define the problem
X = zeros(l,l);
for i = 1 : l
  for j = 1 : l
    X(i,j) = x(i)*x(j);
  end
end
Q = [ X -X ; -X X ];
c = epsilon*ones(2*l,1) + [-y;y];
sol = quadprog(Q,c,[],[],[ones(1,l) -ones(1,l)],0,zeros(2*l,1),C*ones(2*l,1)); % solve the problem
lap = sol(1:l);
lam = sol(l+1:2*l);
w = (lap-lam)'*x ; % compute w
% compute b
ind = find(lap > 10^(-3) & lap < C-10^(-3));
if isempty(ind)==0 %~isempty(ind)
  i = ind(1);
  b = y(i) - w*x(i) - epsilon ;
else
  ind = find(lam > 10^(-3) & lam < C-10^(-3));
  i = ind(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 ;
% 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')
```

nonlinear regression - dual problem (Model)

```
data = [...];
x = data(:,1) ;
y = data(:,2) ;
l = length(x) ; % number of points
epsilon = 10 ; C = 10;
% define the problem
X = zeros(l,l);
for i = 1 : l
   for j = 1 : l
      X(i,j) = kernel(x(i),x(j)) ;
   end
end
Q = [ X -X ; -X X ];
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)==0
   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
z = zeros(l,1); % find regression and epsilon-tube
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 ;
% 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')
```

nonlinear regression - dual problem

```
%% kernel function
function v = kernel(x,y)
   p = 4 ;
   v = (x'*y + 1)^p;
end
```

K-means Clustering (Model)

```
   data = [...];
   l = size(data,1); % number of patterns (punti)
   k=3; %%numero centroidi
   InitialCentroids=[5,7;6,3;4,4]; %%inserire centroidi iniziali qui ((5:7) è il primo, (6:3) il secondo ecc..)
   [x,cluster,v] = kmeans1(data,k,InitialCentroids)
   plot(x(1,1),x(1,2),'b*',x(2,1),x(2,2),'r*',x(3,1),x(3,2),'g*'); % plot centroids
   hold on
   % 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');
function [x,cluster,v] = kmeans1(data,k,InitialCentroids);
   l = size(data,1); % number of patterns
   x = InitialCentroids; % initialize centroids
   cluster = zeros(l,1); % initialize 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
   vold = 0;
   for i = 1 : l % compute the objective function value
      vold = vold + norm(data(i,:)-x(cluster(i),:))^2 ;
   end
   while true
      for j = 1 : k % update centroids
         ind = find(cluster == j);
         if isempty(ind)==0
            x(j,:) = mean(data(ind,:),1);
         end
      end
      for i = 1 : l  % update clustrs
         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
      v = 0;
      for i = 1 : l
         v = v + norm(data(i,:)-x(cluster(i),:))^2 ; % update objective function
      end
```

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

K-median Clustering (Model)

```
  clear; close all; clc;
  data = [ 1.2734    6.2721
    2.7453    7.4345
    1.6954    8.6408
    1.1044    8.6364
    4.8187    7.3664
    4.8462    8.4123];
  l = size(data,1); % number of patterns
  k=3; %%numero centroidi
  InitialCentroids=[5,7;6,3;4,3]; %%mettere centroidi qui
  [x,cluster,v] = kmedian2(data,k,InitialCentroids)
  plot(x(1,1),x(1,2),'b*',x(2,1),x(2,2),'r*',x(3,1),x(3,2),'g*');  % plot centroids
  hold on
  c1 = data(cluster==1,:); c2 = data(cluster==2,:); c3 = data(cluster==3,:); % plot cluster
  plot(c1(:,1),c1(:,2),'bo',c2(:,1),c2(:,2),'ro',c3(:,1),c3(:,2),'go');
function [x,cluster,v] = kmedian2(data,k,InitialCentroids)
  l = size(data,1); % number of patterns
  x = InitialCentroids; % initialize centroids
  cluster = zeros(l,1); % initialize 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
  vold = 0; % compute the objective function value
  for i = 1 : l
    vold = vold + norm(data(i,:)-x(cluster(i),:),1) ;
  end
  while true
    for j = 1 : k   % update centroids
              ind = find(cluster == j);
      if isempty(ind)==0
        x(j,:) = median(data(ind,:),1);
      end
    end
    for i = 1 : l % update clusters
      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
```

K – median (continue)

```
        end
    end
    v = 0;
    for i = 1 : l % update objective function
        v = v + norm(data(i,:)-x(cluster(i),:),1) ;
    end
    if vold - v < 1e-5 % stopping criterion
        break
    else
        vold = v;
    end
  end
end
```

K-means/median Multistart approach (Model)

```
%inserire al posto di: InitialCentroids=[5,7;6,3;4,3]; %%mettere centroidi qui
                                        [x,cluster,v] = kmedian2(data,k,InitialCentroids)
vbest= inf;
xbest=[];
clusterbest=[];
maxiter=1000;
iter=0;
while iter<maxiter
InitialCentroids= 10*rand(k,2);
[x,cluster,v]= kmeans(data,k,InitialCentroids) %o kmedian a seconda di ciò che stai facendo
If v< vbest
xbest=x;
clusterbest=cluster;
vbest=v
end
iter=iter+1
end
%alla fine puoi vedere x e v scrivendo xbest e vbest sul terminale
```