Practical session 1: Linear SVM for two class separable data
(in the Primal)

Stéphane Canu
scanu@insa-rouen.fr, asi.insa-rouen.fr\~scanu

20-25 of july 2015, USP, São Paulo

Practical session description

This practical session aims at writing two functions solving the separable two classes classification problem with linear Support Vector Machines (SVM) in the primal as a quadratic program and the associated linear programing SVM for variable selection. To make it work, you are supposed to have CVX installed (you can download it from `http://cvxr.com/cvx/`)

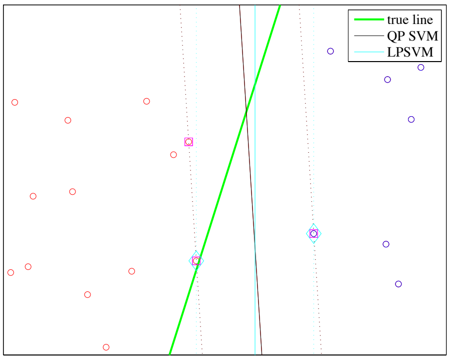

Figure 1: result of TP 1

Ex. 1 — Linear SVM for two class separable data (in the Primal)

1. Build the training data.

   a) Generate a set of 20 data points in dimension 2, uniformly distributed in the square (0,4) and display a scatter plot of this data using red circle.

```
n = 20;             % sample size up to 200000 !
rand('seed',2);     % fix the randomess
Xi = 4*rand(n,2);   % build the training set
q = 0;              % add useless variables
Xi = [Xi 4*rand(n,q)];
[n,p] = size(Xi);
plot(Xi(:,1),Xi(:,2),'or');
```

   b) To make this data set linearly separable, set the labels to 1 for the points above the separating line $\mathrm{w}^\top \mathrm{x} + b = 0$ with $\mathrm{w} = (4, -1)$ and $b = -6$, and -1 for the points under the separating line. This will be your training set. Plot the training set, using red circle for class 1 data points and blue circles for the others.

```
bt = -6;            % define the separation line bias
wt = [4 ; -1];      % define the separation line vector
yi = sign(wt(1) * Xi(:,1) + wt(2) * Xi(:,2) + bt);
hold on; plot(Xi(find(yi==1),1),Xi(find(yi==1),2),'ob');
```

   c) Draw separating line $\mathrm{w}^\top \mathrm{x} + b = 0$ (in green to get figure 1).

```
x1 = 0;
y1 = (-bt-(wt(1)*x1))/wt(2);
x2 = 4;
y2 = (-bt-(wt(1)*x2))/wt(2);
plot([x1 x2],[y1 y2],'g','LineWidth',2)
```

2. Max margin SVM

 a) Using CVX, give a Matlab code for solving

$$\begin{cases} \max\limits_{m,\mathrm{v},a} & m \\ \text{with} & y_i(\mathrm{v}^\top \mathrm{x}_i + a) \geq m \; ; \quad i = 1, n \\ \text{and} & \|\mathrm{v}\|^2 = 1 \end{cases}$$

```
cvx_begin   % The indentation is used for purely stylistic reasons and is optional.
  variables v(p) a m
  maximize( m )
  subject to
    yi.*(Xi*v + a) >= m;
    v'*v <= 1;
cvx_end
```

 b) How long does it takes? (use tic/toc matlab instructions)

 c) Find the indices of the support vectors, and count them

```
vec_sup = find(yi.*(Xi*v + a) <= m+eps^.3);
length(vec_sup)
```

 d) Draw the separating hyperplane found by the max margin SVM and the associated margin and support vectors

```
x1 = 0;                         % left bound
y1 = (-a-(v(1)*x1))/v(2);       % the separating hyperplane
z1 = (m-a-(v(1)*x1))/v(2);      % the margin
zm1 = (-m-a-(v(1)*x1))/v(2);    % the other margin
x2 = 4;                         % right bound
y2 = (-a-(v(1)*x2))/v(2);       % the separating hyperplane
z2 = (m-a-(v(1)*x2))/v(2);      % the margin
zm2 = (-m-a-(v(1)*x2))/v(2);    % the other margin
h = plot([x1 x2],[y1 y2],'k','LineWidth',2);
plot([x1 x2],[z1 z2],':k');     % the margin
plot([x1 x2],[zm1 zm2],':k');   % the other margin
plot(Xi(vec_sup,1),Xi(vec_sup,2),'sm','MarkerSize',10);
```

3. Linear SVM minimizing the norm (usual form)

 a) Using CVX, give a matlab code for solving

$$\begin{cases} \min\limits_{\mathrm{w},b} & \frac{1}{2}\|\mathrm{w}\|^2 \\ \text{with} & y_i(\mathrm{w}^\top \mathrm{x}_i + b) \geq 1 \; ; \quad i = 1, n \end{cases}$$

```
cvx_begin
   variables w(p) b
   minimize( .5*w'*w )
   subject to
       yi.*(Xi*w + b) >= 1;
cvx_end
```

 b) Check that the results given by the max margin and the min norm SVM are the same *i.e.*

$$\mathrm{v} = \frac{\mathrm{w}}{\|\mathrm{w}\|}, \mathrm{v} = m\mathrm{w} \qquad \text{and} \qquad a = \frac{b}{\|\mathrm{w}\|}, a = mb$$

```
[v w/norm(w) w v/m]
[a b/norm(w) b a/m]
```

4. SVM and quadratic programming
   a) Rewrite the min norm SVM problem as a quadratic program in its stand at form and use `quadprog` or `cplexqp` to solve it

```
% X = QUADPROG(H,f,A,b) to solve the quadratic programming problem:
% min 0.5*x'*H*x + f'*x subject to: A*x <= b
% x
```

```
H = [eye(p)];
H(p+1,p+1) = 0;
f = zeros(p+1,1);
A = -[diag(yi)*Xi yi];
bb = -ones(n,1);
x = quadprog(H,f,A,bb);
```

   b) Check that the results provided by CVX and `quadprog` are the same

```
[x [w;b]]
```

   c) How long does it takes. Is it slower or faster than CVX (and why)?

5. Linear programing SVM minimizing the L_1 norm (LP SVM)
   a) Using CVX, give a matlab code for solving

$$\begin{cases} \min\limits_{\mathrm{w},b} & \|\mathrm{w}\|_1 = \sum_{j=1}^{p} |w_j| \\ \text{with} & y_i(\mathrm{w}^\top \mathrm{x}_i + b) \geq 1 \; ; \quad i = 1, n \end{cases}$$

```
cvx_begin
  variables wl(p) bl
  minimize( sum(abs(wl)) )
  subject to
      yi.*(Xi*wl + bl) >= 1;
cvx_end
```

   b) Is it performing variable selection?
   c) Draw the separating line estimated by LP SVM

```
vec_sup = find(yi.*(Xi*wl + bl) < 1 + eps^.3);
plot(Xi(vec_sup,1),Xi(vec_sup,2),'dc','MarkerSize',15);
x1 = 0;
y1 = (-bl-(wl(1)*x1))/(wl(2)+eps^.5);
z1 = (1-bl-(wl(1)*x1))/wl(2);
zm1 = (-1-bl-(wl(1)*x1))/wl(2);
x2 = 4;
y2 = (-bl-(wl(1)*x2))/(wl(2)+eps^.5);
z2 = (1-bl-(wl(1)*x2))/wl(2);
zm2 = (-1-bl-(wl(1)*x2))/wl(2);
plot([x1 x2],[y1 y2],'c')
plot([x1 x2],[z1 z2],':c')
plot([x1 x2],[zm1 zm2],':c')
```

   d) Rewrite the LP SVM problem as a linear program in its standart form and use `linprog` or `cplexlp` to solve it

```
% X = linprog(f,A,b,Aeq,beq,LB,UB) attempts to solve the linear programming problem:
% min f'*x subject to: A*x <= b  and  Aeq*x = beq
% x
% so that the solution is in the range LB <= X <= UB
```

```
f = [ones(2*p,1); 0];
A = [-diag(yi)*Xi diag(yi)*Xi -yi];
bb = -ones(n,1);

xl = linprog(f,A,bb,[],[],[zeros(2*p,1); -inf]);
wlp = xl(1:p) - xl(p+1:2*p);
blp = x(end);
```

e) Check that the results are the same and compare computing time.

```
[[wl ; bl] [wlp ; blp]]
```

6. Compare all the results and computing time. Produce figure 1.

```
[[wt ; bt] [w ; b] x [wl ; bl]]
```

7. Write two matlab functions `SVMClassPrimal, SVMValPrimal` for solving the separable two classes classification problem with linear Support Vector Machines (SVM) in the primal as a quadratic program and the associated linear programing SVM for variable selection.

```
[w,b] = SVMClassPrimal(Xi,yi,opt);
% opt = 1 for LP SMV and opt = 2 for QP SVM
% you may also ofer the possibility for the user too choose the solver

[y_pred] = SVMValPrimal(Xtest,w,b);
```

My solution for $n = 2000$ and $q = 200$

```
QP SVM
    CVX for max margin SVM: 6.1186
   Number of support vectors: 197
      CVX for min norm SVM: 7.0278
 quadprog for min norm SVM: 5.3852
  cplexqp for min norm SVM: 1.2188
LP SVM
       CVX for L1 norm SVM: 11.9319
   linprog for L1 norm SVM: 19.1131
   cplexlp for L1 norm SVM: 0.8705
```