Practical session : Linear SVM for two class separable data

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

september the 9th 2014, Ocean's Big Data Mining, Brest

Practical session description

This practical session aims at writing two functions solving the separable two classes classification problem with linear Support Vector Machines (SVM) as a quadratic program in different situations: primal dual, without and with noise. To make it work, you are supposed to have CVX installed (you can download it from `http://cvxr.com/cvx/`) as well as the `quadprog` matlab function available in the matlab optimization toolbox.

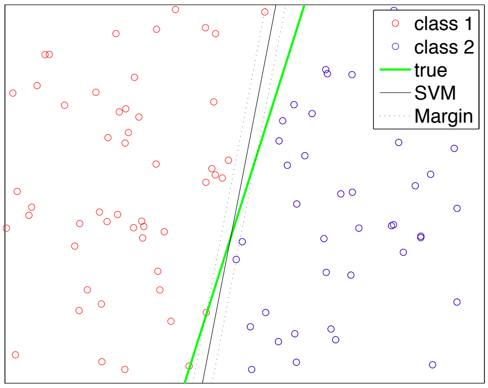

Figure 1: result of TP 1

Ex. 1 — Linear SVM for two class separable data

1. Generate a set of 100 data points in dimension 2, uniformly distributed in the square (0,4). 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 $b = -6$. This will be your training set.

```
n = 100;                    % 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 to see what... up to 180;
Xi = [Xi 4*rand(n,q)];
[n,p] = size(Xi);

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);
```

2. Plot the training set, using red circle for class 1 data points and blue circles for the others. Draw separating line $\mathrm{w}^\top \mathrm{x} + b = 0$ with $\mathrm{w} = (4, -1)$ and $b = -6$ (in green to get figure 1).

```
plot(Xi(:,1),Xi(:,2),'or');
hold on
plot(Xi(find(yi==1),1),Xi(find(yi==1),2),'ob');
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)
```

3. 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
   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

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

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

```
x1 = 0;
y1 = (-a-(v(1)*x1))/v(2);
z1 = (m-a-(v(1)*x1))/v(2);
zm1 = (-m-a-(v(1)*x1))/v(2);
x2 = 4;
y2 = (-a-(v(1)*x2))/v(2);
z2 = (m-a-(v(1)*x2))/v(2);
zm2 = (-m-a-(v(1)*x2))/v(2);
plot([x1 x2],[y1 y2],'r')
plot([x1 x2],[z1 z2],':r')
plot([x1 x2],[zm1 zm2],':r')
plot(Xi(vec_sup,1),Xi(vec_sup,2),'sm','MarkerSize',10);
```

4. 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
   dual variables pi
   minimize( .5*w'*w )
   subject to
      pi :  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]
```

5. Write the KKT condition associated wight the solution
   a) Based on the previous results (w, b), retrieve the active set (the indices of support vectors)

```
A = find(yi.*(Xi*w + b) == 1);
A = find(yi.*(Xi*w + b) <= 1.00001);
cA = length(A);
```

   b) Write the KKT system of equation

```
DA = diag(yi(A));

KKT = [eye(p)          -Xi(A,:)'*DA      zeros(p,1)
          -DA*Xi(A,:)   zeros(cA)      -yi(A)
          zeros(1,p)     -yi(A)'            zeros(1)];
Kb = [zeros(p,1) ; -ones(cA,1) ;0];

sol = KKT\Kb;
```

   c) Check that the solution provided by matlab and the one given by solving the KKT are the same

```
[[w;pi(A);b] sol]
```

6. 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)?

7. Max Margin SVM in the dual
   a) Using CVX, give a matlab code for solving Max Margin SVM in the dual

$$\begin{cases} \min\limits_{\alpha} & \frac{1}{2}\,\alpha^\top G\alpha - \sum_i \alpha_i \\ \text{with} & \alpha^\top y_i = 0\,; \\ \text{and} & 0 \leq \alpha_i\,; \qquad\qquad i = 1, n \end{cases}$$

```
G = (yi*yi').*(Xi*Xi');
e = ones(n,1);
cvx_begin
   variable a(n)
   dual variables de dp
   minimize( 1/2*a'*G*a - e'*a )
   subject to
      de : yi'*a == 0;
      dp :  a >= 0;
cvx_end
```

b) Check that the dual variable of the primal are the same as the variables of the dual

```
[a pi]
```

c) Check that the dual variable of the primal are the same as the variables of the dual

```
[b de]
```

d) Using the representer theorem (KKT for stationarity with respect to w), recompute w using the dual variables

```
[w  Xi'*(yi.*a)]
```

8. Using `quadprog` to solve both primal and dual SVM formulations

a) Modify the outputs of the `quadprog` you wrote for solving the min norm SVM problem to get the dual variables (the Lagrange multipliers)

```
% [X,FVAL,EXITFLAG,OUTPUT,LAMBDA] = QUADPROG(H,f,A,b) returns the set
%    Lagrangian multipliers LAMBDA, at the solution: LAMBDA.ineqlin
%    linear inequalities A, LAMBDA.eqlin for the linear equalities Aeq
%    LAMBDA.lower for LB, and LAMBDA.upper for UB.H = [eye(p)];
```

```
H(p+1,p+1) = 0;
f = zeros(p+1,1);
A = -[diag(yi)*Xi yi];
bb = -ones(n,1);
[xp, VAL,EXITFLAG,OUTPUT,lambda] = quadprog(H,f,A,bb);
```

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

```
l = eps^.5;
G = G + l*eye(n);    % 7) the secret to make it work
tic
ad = quadprog(G,-e,[],[],yi',0,zeros(n,1),inf*ones(n,1));
```

c) Download and instal the SVMKM toolbox from
`http://asi.insa-rouen.fr/enseignants/~arakoto/toolbox/`.
Solve the same min norm SVM dual problem using the `monqp` solver included in the SVMKM toolbox.

```
% function [xnew, lambda, pos] = monqp(H,c,A,b,C,l,verbose,X,ps,xinit)
%
% min 1/2  x' H x - c' x
%    x
%   contrainte    A' x = b
%
%   et           0 <= x_i   <= C_i
[alpha, b, pos] = monqp(G,e,yi,0,inf,l,0);
```

d) Using the output of `monqp`, recompute the whole dual variables and the associated primal variables.

```
aqp = zeros(n,1);
aqp(pos) = alpha;
wqp = Xi(pos,:)'*(yi(pos).*alpha);
```

9. Compare all the results and computing time.
10. Write two matlab functions `SVMClass, SVMVal` for solving the separable two classes classification problem with linear Support Vector Machines (SVM) in the primal as a quadratic program.

```
[w,b] = SVMClass(Xi,yi,opt);
%  opt for some options
%  you may also ofer the possibility for the user too choose the solver

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