DM_2 – Practical session: SVM as a kernel machine

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

March 2, 2016

Practical session description

This practical session aims at at coding the C-SVM with kernels. You'll see that your implementation will have very few difference with the one you have done in the linear case. To make it work, you are supposed to have CVX installed (you can download it from `cvxr.com/cvx/`) as well as the SVMKM toolbox (`asi.insa-rouen.fr/enseignants/~arakoto/toolbox/`).

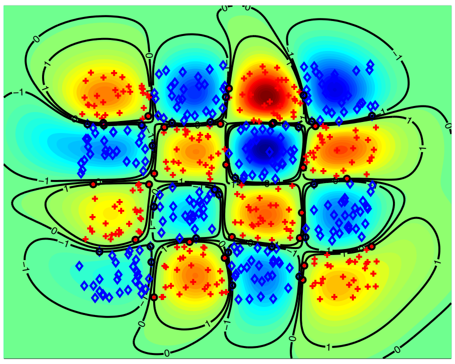

Figure 1: Result of practical session 1: an example SVM at work with kernel

Ex. 1 — SVM with kernel

1. Generate a set of 500 data points in dimension 2, from two classes using the checker problem. You can generate this data using the `dataset_KM` matlab function available on Moodle. This will be your training set. You can also download it from Moodle. visualize your data.

```
n = 500; % upto n = 10000;
[Xapp,yapp,Xtest,ytest]=dataset_KM('checkers',n,n^2);
[n,p] = size(Xapp);

figure(1); hold on
h1=plot(Xapp(yapp==1,1),Xapp(yapp==1,2),'+r','LineWidth',2);
h2=plot(Xapp(yapp==-1,1),Xapp(yapp==-1,2),'db','LineWidth',2);
```

2. Solving the C-SVM with gaussian kernel
   a) Compute a gaussian kernel and the matrix on your data with kerneloption = .5.

```
D = (Xapp*Xapp');
N = diag(D);
D = -2*D + N*ones(1,n) + ones(n,1)*N';
kerneloption = .5;
s = 2*kerneloption^2;
K = (exp(-D/s));
G = (yapp*yapp').*K;
```

   b) Compute the same gaussian kernel using the `svmkernel` function of the SVMKM toolbox

```
kernel = 'gaussian';
kerneloption = .5;

K=svmkernel(Xapp,kernel,kerneloption);
G = (yapp*yapp').*K;
```

c) Adapt you CVX code for solving the C-SVM with a gaussian kernel (C = 10000)

```
C = 10000;
e = ones(n,1);
cvx_begin
   variable a(n)
   dual variables de dp dC
   minimize( 1/2*a'*G*a - e'*a )
   subject to
      de : yapp'*a == 0;
      dp : a >= 0;
      dC : a <= C;
cvx_end
```

d) Solve the same problem using `monqp`

```
lambda = eps^.5;
[alpha,b,pos] = monqp(G,e,yapp,0,C,lambda,0);
```

e) Compare all the results and computing time.

3. Do the nice plot

a) type `help mesh grid` in the matlab workspace to see what it is about.

b) Generate a grid $[-1 : .01 : 1] * 3, [-1 : 0.01 : 1] * 3$

```
[xtest1 xtest2] = meshgrid([-1:.01:1]*3,[-1:0.01:1]*3);
```

c) Based on that grid, generate a two diminutional test vector, its associated kernel values with the support vectors and compute the predicted output of the SVM.

```
nn = length(xtest1);
Xgrid = [reshape(xtest1 ,nn*nn,1) reshape(xtest2 ,nn*nn,1)];
Kgrid = svmkernel(Xgrid,kernel,kerneloption,Xapp(pos,:));
ypred = Kgrid*(yapp(pos).*alpha) + b;
```

d) reshape the predicted output of the SVM on the initial grid

```
ypred = reshape(ypred,nn,nn);
```

e) Do the plot

```
contourf(xtest1,xtest2,ypred,50); shading flat;
hold on;
[cc,hh]=contour(xtest1,xtest2,ypred,[-1 0 1],'k');
clabel(cc,hh);
set(hh,'LineWidth',2);
h1=plot(Xapp(yapp==1,1),Xapp(yapp==1,2),'+r','LineWidth',2);
h2=plot(Xapp(yapp==-1,1),Xapp(yapp==-1,2),'db','LineWidth',2);
xsup = Xapp(pos,:);
h3=plot(xsup(:,1),xsup(:,2),'ok','LineWidth',2);
axis([-3 3 -3 3]);
```

4. Solve the same problem with $n = 10000$ (DON'T USE CVX it doesn't scale that much)

5. Solve the same problem back with $n = 500$ but with the bandwidth (kerneloption) set to 0.05. WARNING: it may be a bit long...

6. Try the polynomial kernel of degree p on some other data

```
K = (Xapp*Xapp' + ones(n)).^p;
```

7. Code $L2$ SVM on the same example using both CVX and `monqp`.

8. Write two matlab functions `MySVMClass, MySVMVal` for solving the two classes classification problem with kernelize Support Vector Machines (SVM) in the dual as a quadratic program.

```
[Xsup,alpha,b] = MySVMClass(Xi, yi, C, kernel, kerneloption, options);
[y_pred] = MySVMVal(Xtest,Xsup,alpha,b,kernel, kerneloption);
```