Pineau Edouard
Master MASH

Homework 3

Question 1

- Lagrange function:

$$L(w,z;\lambda,\pi) = \frac{1}{2}w^T w + C\mathbf{1}^T z + \sum_{i=1}^{n}\lambda_i(1 - y_i(w^T x_i) - z_i) - \pi^T z$$

- Dual function

$$g(\lambda,\pi) = \inf_{w,z} L(w,z;\lambda,\pi) = \inf_{w,z}\frac{1}{2}w^T w + C\mathbf{1}^T z + \sum_{i=1}^{n}\lambda_i(1 - y_i(w^T x_i) - z_i) - \pi^T z$$

We respectively optimize over w and z:

- $\partial_w L(w,z;\lambda,\pi) = w - \sum_{i=1}^{n}\lambda_i y_i x_i = 0 \rightarrow w = \sum_{i=1}^{n}\lambda_i y_i x_i \in \mathbb{R}^d$
- We see that L is unbounded below if $C\mathbf{1} - \lambda - \pi \neq 0 \rightarrow \lambda = C\mathbf{1} - \pi$

Hence:

$$g(\lambda,\pi) = L\left(\sum_{i=1}^{n}\lambda_i y_i x_i\,, z; \lambda, C\mathbf{1} - \lambda\right) = \lambda^T\mathbf{1} - \frac{1}{2}\left\|\sum_{i=1}^{n}\lambda_i y_i x_i\right\|_2^2 \text{ with } \lambda = C\mathbf{1} - \pi \succcurlyeq 0, \pi \succcurlyeq 0$$

- Dual problem:

$$\begin{cases} \max\limits_{\lambda,\pi} \lambda^T\mathbf{1} - \frac{1}{2}\left\|\sum\limits_{i=1}^{n}\lambda_i y_i x_i\right\|_2^2 \\ s.t.\ \lambda \succcurlyeq 0, \pi \succcurlyeq 0, \lambda = C\mathbf{1} - \pi \end{cases}$$

We observe that the constraints of the dual problem are equivalent to $0 \preccurlyeq \lambda \preccurlyeq C\mathbf{1}, \sum_{i=1}^{n}\lambda_i y_i = 0$ and that we can erase the variable π. Hence the dual problem is:

$$\begin{cases} \max\limits_{\lambda} \lambda^T\mathbf{1} - \frac{1}{2}\left\|\sum\limits_{i=1}^{n}\lambda_i y_i x_i\right\|_2^2 \\ s.t.\ 0 \preccurlyeq \lambda \preccurlyeq C\mathbf{1}, \sum\limits_{i=1}^{n}\lambda_i y_i = 0 \end{cases}$$

Question 2

For primal, the strict feasible set corresponds to:

$$dom(f) \cap \{(w,z) \in \mathbb{R}^{dn} : y_i(w^T x_i) + z_i > 1 \; \forall i = 1 \dots n, z \succ 0\}$$

For the dual program, the strict feasible set is:

$$dom(g) \cap \{(\lambda,\pi) \in \mathbb{R}^{2n} : \lambda \succ 0, \pi \succ 0, \lambda = C\mathbf{1} - \pi\}$$

Question 3

- Generalization of the SVM program from

For the primal:

We want to transform:

- $\frac{1}{2}w^T w + C\mathbf{1}^T z$ into $\frac{1}{2}x^T Q x + p^T x$
- $y_i(w^T x_i) - z_i \geq 1, z \geq 0$ into $Ax - b \preccurlyeq 0$

By identification:

$$\boxed{Q = \begin{pmatrix} I_d & 0_{d\times n} \\ 0_{n\times d} & 0_{n\times n} \end{pmatrix}, p = \begin{pmatrix} 0_d \\ C\mathbf{1}_n \end{pmatrix}, A = \begin{pmatrix} -diag(y)X & -I_n \\ 0_{n\times n} & -I_n \end{pmatrix}, b = \begin{pmatrix} \mathbf{1_n} \\ 0_n \end{pmatrix}}$$

```
function [Q,p,A,b] = transform_svm_primal(C,X,y)
[n,d]=size(X);

%objective function
Q=[eye(d),zeros(d,n);zeros(n,d),zeros(n,n)];
p=[zeros(d,1);C*ones(n,1)];

%constraints
A11=-y.*X;
A12=-eye(n);
A21=zeros(n,d);
A22=-eye(n);
A=[A11,A12;A21,A22];
b=[-ones(n,1);zeros(n,1)];
end
```

For the dual:

We have:

$$\lambda^T\mathbf{1} - \frac{1}{2}\left\|\sum_{i=1}^{n}\lambda_i y_i x_i\right\|_2^2 = \lambda^T\mathbf{1} - \frac{1}{2}\lambda^T diag(y)XX^T diag(y)\lambda$$

Moreover, thee dual problem is a maximization problem, we can transform it like:

$$\begin{cases} -\min\limits_{\lambda} -\lambda^T\mathbf{1_n} + \frac{1}{2}\lambda^T diag(y)XX^T diag(y)\lambda \\ s.t.\; 0 \preccurlyeq \lambda \preccurlyeq C\mathbf{1} \end{cases}$$

We want to transform

- $-\lambda^T \mathbf{1_n} + \frac{1}{2}\lambda^T diag(y) X X^T diag(y) \lambda$ into $\frac{1}{2} x^T Q x + p^T \mathbf{1_n}$
- $0 \leqslant \lambda \leqslant C\mathbf{1_n}$ into $Ax - b \leqslant 0$

We have $0 \geqslant -\lambda$ and $\lambda \leqslant C\mathbf{1_n}$, therefore:

$$\boxed{A = \begin{pmatrix} -I_n \\ I_n \end{pmatrix}, b = (0_n, C\mathbf{1}_n)}$$

By identification:

$$\boxed{Q = diag(y) X X^T diag(y), p = -1_n}$$

Then:

```
function [Q,p,A,b] = transform_svm_dual(C,X,y)
[n,d]=size(X);
%objective function
Q=diag(y)*X*X'*diag(y);
p=-ones(n,1);
%constraints
A=[-eye(n);eye(n)];
b=[zeros(n,1);C*ones(n,1)];
end
```

- Newton's step
    - Backtracking line search:

```
function[step_size_new]=BTLS(x,t,Q,p,A,b,step,df)
%parameters of the backtracking linesearch
step_size=1;
alpha=1/2;
beta=0.9;

%function value before and after ste
f_origin=f(t,x,Q,p,A,b);
f_step=f(t,x+step_size*step,Q,p,A,b);

while(f_step>=f_origin+alpha*step_size*transpose(df)*step ...
        || sum(A*(x+step_size*step)-b>0)>0)
    step_size=step_size*beta;
    f_step=f(t,x+step_size*step,Q,p,A,b);
end

step_size_new=step_size;
end
```

The introduction of a feasibility condition within the while loop enable to never get out the feasible set.

- Newton's step using backtracking line search:

```
function [x_star,gap] = Newton(t,x,Q,p,A,b)
df=grad(t,x,Q,p,A,b);
```

```
hf=hes(t,x,Q,p,A,b);
step=-hf\df;
%step=-inv(hf)*df;
step_size=BTLS(x,t,Q,p,A,b,step,df);

%we have to verify that the point x is into the domain
if(sum(A*x-b>0)>0)
    fprintf('Failure, point out of the domain');
    x_star=[];gap=[];
    return;
end

%performance of one Newton step
x_star=x+step*step_size;
gap=(1/2)*(transpose(step)*hf*step);
end
```

The functions `grad(t,x,Q,p,A,b)` and `hes(t,x,Q,p,A,b)` are built explicitly for the quadratic problem:

```
function[y]=grad(t,x,Q,p,A,b)
y_temp=1./(A*x-b);
y=t*(Q*x+p)-A'*y_temp;
end

function[y]=hes(t,x,Q,p,A,b)
y_temp=1./(A*x-b);
y=t*Q+A'*diag(y_temp.^2)*A;
end
```

- Centring step using Newton's method

We use the Newton step function iteratively until the proximity of x to x^* is below the tolerance. This proximity is calculated with the formula of the Newton decrement:

$$\lambda(x) = \left(\nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x)\right)^{1/2}$$

This value is the `gap` output by the function Newton.

```
function [x_star,x_seq] = centering_step(t,x,Q,p,A,b,tol)
[x_star,gap]=Newton(t,x,Q,p,A,b);
x_seq=x_star;
while(gap>tol)
    [x_star,gap]=Newton(t,x_star,Q,p,A,b);
    x_seq=[x_seq x_star];
end
end
```

- Implementation of barrier method

```
function[x_sol,x_seq] = barr_method(Q,p,A,b,x_0,mu,tol)
%initialization
m=length(x_0);
t=1;
x_star=centering_step(t,x_0,Q,p,A,b,tol);
```

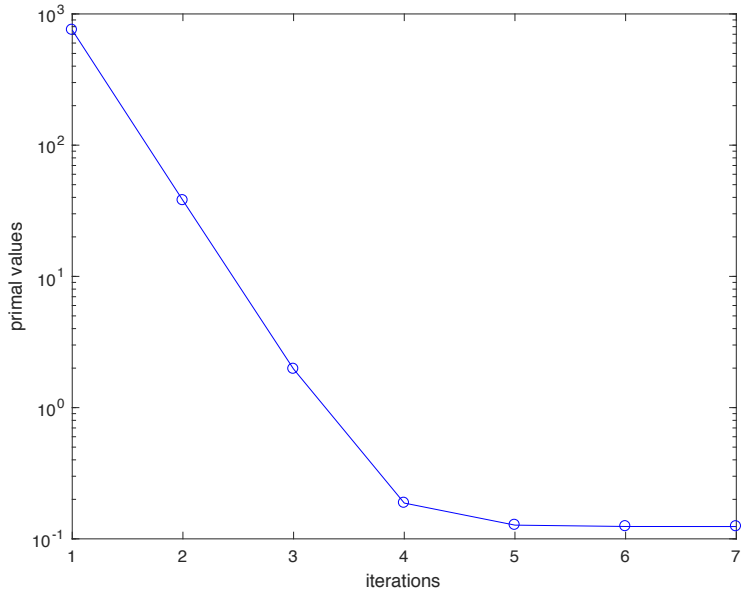
```
x_seq=x_star;
t=mu*t;
%barrier method
while(m/t>tol)
    x_star=centering_step(t,x_star,Q,p,A,b,tol);
    t=mu*t;
    x_seq=[x_seq,x_star];
end
x_sol=x_star;
end
```

Remark: for the following questions, the plots have been maid with:

$$d = 2, n = 500, \mu_1 = \begin{pmatrix}5\\5\end{pmatrix}, \mu_2 = \begin{pmatrix}-5\\-5\end{pmatrix}, \Sigma_1 = \begin{pmatrix}4 & 0\\0 & 4\end{pmatrix}, \Sigma_2 = \begin{pmatrix}3 & 0\\0 & 3\end{pmatrix}, C = 1, tol = 10e^{-6}$$

Question 4

We use the sequence of $(x_i)_{i=1\dots n}$ created during the barrier method to build our graph for the primal evolution at each iteration.



with the code:

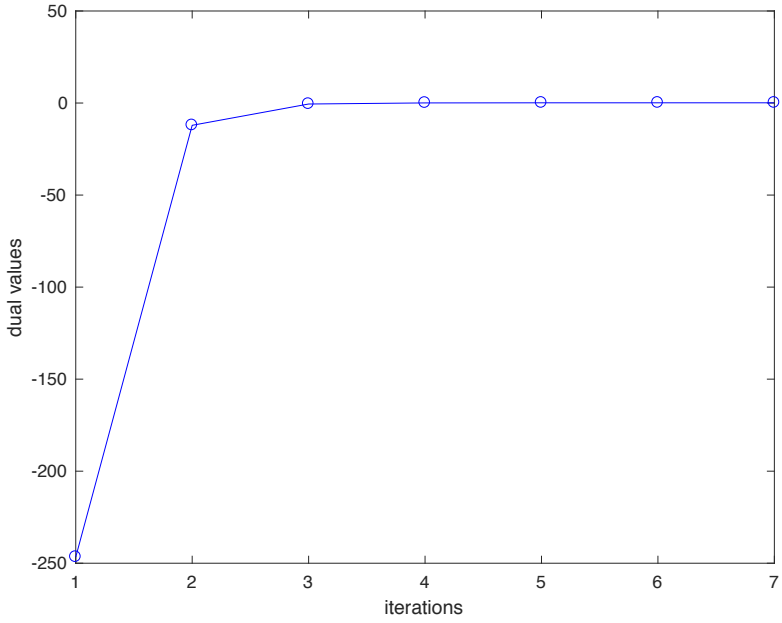
```
function[primal_evolution]=plot_primal(x_seq,Q,p)
    primal_evolution=[1 quadratic_obj(x_seq(:,1),Q,p)];
    for i=2:size(x_seq,2)
        primal_evolution=[primal_evolution;[i ...
            quadratic_obj(x_seq(:,i),Q,p)]];
    end

    figure
    semilogy(primal_evolution(:,1),primal_evolution(:,2),'bo-')
    xlabel('iterations')
    ylabel('primal values')
end
```

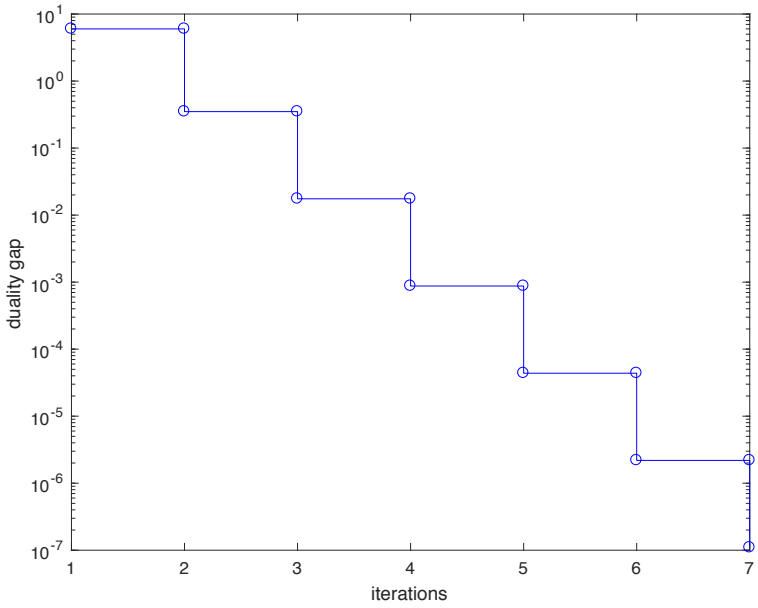
Plotting:

```
[x_star_primal,x_seq_primal] = ...
    barr_method(Q_primal,p_primal,A_primal,b_primal,x_0_primal,mu,tol);
primal_evol=plot_primal(x_seq_primal,Q_primal,p_primal)
```

Question 5

We use the sequence of $(x_i)_{i=1...n}$ created during the barrier method to build our graph for the dual evolution at each iteration.





with the code:

```
function[dual_evolution]=plot_dual(x_seq,Q,p,mu)
dual_evolution=[1 quadratic_obj(x_seq(:,1),Q,p)];
t=1;
duality_gap=[t,6];
t=t*mu;
for i=2:size(x_seq,2)
    dual_evolution=[dual_evolution;...
        [i quadratic_obj(x_seq(:,i),Q,p)]];
    duality_gap=[duality_gap;[i size(x_seq,2)/t]];
    t=t*mu;
end

%plot of the dual objective value at each iteration
figure
plot(dual_evolution(:,1),-dual_evolution(:,2),'bo-')
xlabel('iterations')
ylabel('dual values')

%plot of duality gap at each iteration
figure
[xx, yy] = stairs(duality_gap(:,1),duality_gap(:,2));
semilogy(xx,yy,'bo-')
xlabel('iterations')
ylabel('duality gap')

end
```

Plotting:

```
[x_star_dual,x_seq_dual] = ...
    barr_method(Q_dual,p_dual,A_dual,b_dual,x_0_dual,mu,tol);
dual_evol=plot_dual(x_seq_dual,Q_dual,p_dual,mu)
```

The convergence of the duality gap to 0 confirms the theoretical result saying that a convex problem with affine hence convex inequality constraints with existence of a strictly feasible point implies strong duality. In our example with the data generated for the case, we have:

$$p^* = d^* \approx 0.1241$$

Question 6

- Creation of artificial data

```
function[X1,y1,X2,y2]=generate_data(n,mu1,mu2,sigma1,sigma2)
X1=mvnrnd(mu1,sigma1,round(n/2));
y1=ones(round(n/2),1);
X2=mvnrnd(mu2,sigma2,n-round(n/2));
y2=-ones(n-round(n/2),1);
end
```

- Optimal linear classifier

The optimal classifier is defined as:

$$g: x \to \begin{cases} 1 \; if \; w^T x + b > 0 \\ -1 \; otherwise \end{cases}$$

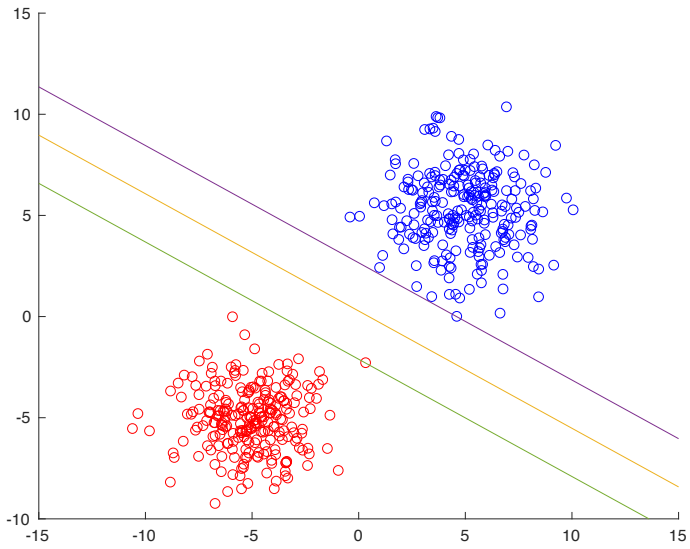
with $b \in \mathbb{R}$ defined such that $\forall i \in [\![1:n]\!], z_i = y_i b$.

Hence we can build a linear classifier that first centres the data (to avoid the addition of a

```
function[y]=linear_classifier(X,w)
% to make sure that data set is in a good shape
if size(X,1)==length(w)
    X=X';
end
% thanks to the addition of an intercept, we do not have to bother
the b
y=X*w./abs(X*w);
end
```

- Plot of the clouds of points and support vectors

The means and variance matrices have been chosen such that the dataset is linearly separable with high probability, this to facilitate de representation. In the last point of the question, this assumption will be erased.



with the code:

```
scatter(X1(:,1),X1(:,2),'blue')
hold on
scatter(X2(:,1),X2(:,2),'red')
fplot(@(x) -x_star_primal(1)/x_star_primal(2)*x-...
    x_star_primal(3)/x_star_primal(2))
fplot(@(x) -x_star_primal(1)/x_star_primal(2)*x-...
```

```
    x_star_primal(3)/x_star_primal(2)+1/x_star_primal(2))
fplot(@(x) -x_star_primal(1)/x_star_primal(2)*x-...
    x_star_primal(3)/x_star_primal(2)-1/x_star_primal(2))
hold off
```

The solid line represents the maximum margin separating hyperplane. The respectively green and blue lines parallel to the decision boundary (yellow line) are the supporting hyperplanes for each cloud, passing through the points (three points here, two positive and one negative) that have the smallest margins. These three points are the support vectors.

- Influence of C over the results and performance of the method

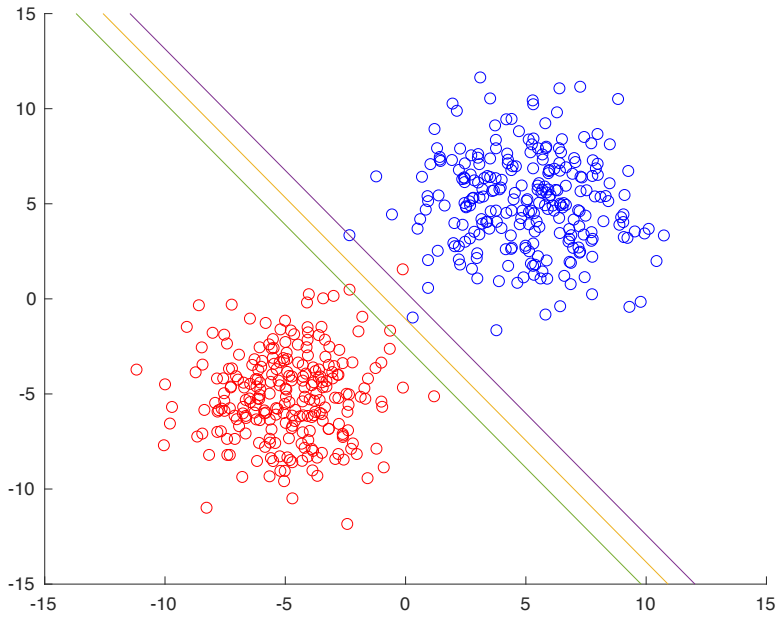
The parameter C will ensure the algorithm still works for non-separable datasets by arbitrating between the two goals of making $\|w\|_2^2$ small and ensure that sufficiently large margins are created (despite the possibility to not optimally classify the train data). The parameter will have an influence over the results and performance of the method.

By construction, the SVM program minimizes $\frac{1}{2}w^T w + C\mathbf{1}^T z$, and equivalently maximises the margins. Considering the constraints, if C takes larger values, the minimum value achieved will be bigger, and equivalently the margins tighter, and conversely. Hence a bigger value of C will encourage better classification of the training data but smaller margins, and conversely.

To illustrate, I chose new covariance matrices, with bigger spread of the distribution:

$$\boxed{\Sigma_1 = \begin{pmatrix} 6 & 0 \\ 0 & 6 \end{pmatrix}, \Sigma_2 = \begin{pmatrix} 5 & 0 \\ 0 & 5 \end{pmatrix}}$$

For C=10:



For C=1: