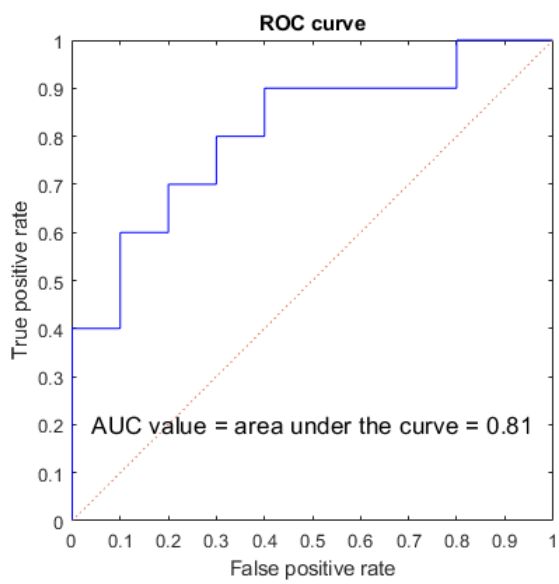
Machine Learning
Homework 2

1. Let the sequence P, P, P, P, N, P, P, N, P, N, P, N, P, N, N, N, N, P, N, N be the sorted result according to the *posterior probability* being a positive instance. Please find the AUC value for this ranking result. (10 %)

   Ans:




```
function ROC_curve_plot(x, y)
    %%% test data
    % x = [0 0 0 0 0 1 1 1 2 2 3 3 4 4 5 6 7 8 8 9 10]/10;
    % y = [0 1 2 3 4 4 5 6 6 7 7 8 8 9 9 9 9 9 10 10 10]/10;
    %%%
    plot (x, y);
    hold on;
    plot([0 1],[0 1],':')
    axis square
    set(gca,'XTick',-0.1:0.1:1.1);
    set(gca,'YTick',-0.1:0.1:1.1);

    title('ROC curve');
    xlabel('False positive rate') ;
    ylabel('True positive rate');
    T = text(0.04,0.2,'AUC value = area under the curve = 0.81','FontSize',13);
end
```

2. **(a)** Solve

$$\min_{\mathbf{x}\in R^2} \ \frac{1}{2}\mathbf{x}^\top \begin{bmatrix} 1 & 0 \\ 0 & 1000 \end{bmatrix} \mathbf{x}$$

using the *steep descent with exact line search*. You are welcome to copy the MATLAB code from my slides. Start your code with the initial point $\mathbf{x}^0 = [1000 \ 1]^\top$. Stop until $\|\mathbf{x}^{n+1} - \mathbf{x}^n\|_2 < 10^{-8}$. Report your solution and the number of iteration. (10 %)

Ans:

```
x =                 f_value =              iter =

   1.0e-05 *         6.2362e-12                  9731

    0.3530
   -0.0004
```

```
function [x, f_value, iter] = grdlines(Q, p, x0, esp)
%%% test data
% Q = [1 0; 0 1000]; p = [0 0]';
% x0 = [1000 1 ]'; esp = power(10, -8);
%%%
% min 0.5 * x'Qx + p'x
% Solving unconstrained minimization via
% steep descent with exact line search
    flag = 1;
    iter = 0;
    while flag > esp
        grad = Q*x0+p;
        temp1 = grad'*grad;
        if temp1 < power(10, -12)
            flag = esp;
        else
            stepsize = temp1/(grad'*Q*grad);
            x1 = x0 - stepsize*grad;
            flag = norm(x1-x0);
            x0 = x1;
        end
            iter = iter + 1;
    end
    x = x0;
    f_value = 0.5*x'*Q*x+p'*x;
end
```

(b) Implement the Newton's method for minimizing a quadratic function $f(x) = \frac{1}{2}\mathbf{x}^\top Q\mathbf{x} + p^\top x$ in MATLAB code. Apply your code to solve the minimization problem in **(a)**.(10 %)

```
x =            f_value =           iter =

     0              0                   2
     0
```

```matlab
function [x, f_value, iter] = Newton_method(Q, p, x0, esp)
%%% test data
% Q = [1 0; 0 1000]; p = [0 0]';
% x0 = [1000 1 ]'; esp = power(10, -8);
%%%
% min 0.5 * x'Qx + p'x
% Solving unconstrained minimization via
% Newton's method
% x1 = x0 - grad/hessian
% grad = Q*x0,  hessian = Q
    flag = 1;
    iter = 0;
    while flag > esp
        grad = Q*x0 + p;
        temp1 = grad'*grad;
        if temp1 < power(10, -12)
            flag = esp;
        else
            x1 = x0 - (x0 + inv(Q)*p);
            flag = norm(x1-x0);
            x0 = x1;
        end
            iter = iter + 1;
    end
    x = x0;
    f_value = 0.5*x'*Q*x+p'*x;
end
```

3. Let $S = \{(\mathbf{x}^i, y_i)\}_{i=1}^{\ell} \subseteq R^n \times \{-1, 1\}$ be a non-trivial training set. The Perceptron Algorithm in *dual form* is given as follows:

Given a training set S

$\alpha \longleftarrow \mathbf{0}$ and $b \longleftarrow 0$
$L \longleftarrow \max_{1 \le i \le \ell} \|\mathbf{x}^i\|_2$
Repeat
for $i = 1$ *to* ℓ
if $y_i(\sum_{j=1}^{\ell} \alpha_j y_j \langle \mathbf{x}^i, \mathbf{x}^j \rangle + b) \le 0$ then
$\alpha_i \longleftarrow \alpha_i + 1$
$b \longleftarrow b + y_i L^2$
end if
end for
end for

until no mistakes made within the *for* loop
return (α, b) and define the linear classifier

$$f(x) = \sum_{i=1}^{\ell} \alpha_i y_i \langle \mathbf{x}^i, \mathbf{x} \rangle + b$$

Suppose that the input training set S is linearly separable.

(a) What are the meanings of the output α_i and the 1-norm of α? (10%)

(b) Why the updating rule is effective? (10%)

Ans: (a)
(1)
$\alpha_i > 0$ means that the training point $(\mathbf{x}^i, y_i)$ has been misclassified in the training process at least once.
$\alpha_i = 0$ means that it has been classified correctly and if removing the training point $(\mathbf{x}^i, y_i)$, it will not affect the final results.
(2)
$\|\alpha_i\|_1$: The number of updates

Ans: (b)
We can apply the Theorem (Novikoff), the number of mistakes made by the on-line perceptron algorithm on S is almost $(\frac{2R}{\gamma})^2$. So we can finish the whole process within finite time.

4. Let $A_+ = \{(0,0),(0.5,0),(0,0.5),(-0.5,0),(0,-0.5)\}$ and
$A_- = \{(0.5,0.5),(0.5,-0.5),(-0.5,0.5),(-0.5,-0.5),(1,0),(0,1),(-1,0),(0,-1)\}$. (50 %)

(a) Try to find the hypothesis $h(\mathbf{x})$ by implementing the Perceptron algorithm in the *dual form* and replacing the inner product

$$< x^i, x^j > \text{ by } < x^i, x^j >^2, \text{ and } L = \max_{1 \le i \le \ell} \|x^i\|_2^2$$

(b) Generate 10,000 points in the box $[-1.5, 1.5] \times [-1.5, 1.5]$ randomly as a test set. Plug these points into the hypothesis that you got in **(a)** and then plot the points for which $h(x) > 0$ with '+'.

(c) Repeat **(a)** and **(b)** by using the training data

$$B_+ = \{(0.5,0),(0,0.5),(-0.5,0),(0,-0.5)\} \text{ and}$$

$$B_- = \{(0.5,0.5),(0.5,-0.5),(-0.5,0.5),(-0.5,-0.5)\}.$$

(d) Let the nonlinear mapping $\phi : R^2 \to R^4$ defined by

$$\phi(\mathbf{x}) = [-x_1x_2,\ x_1^2,\ x_1x_2,\ x_2^2\ ]$$

Map the training data A_+ and A_- into the feature space using this nonlinear map. Find the hypothesis $f(x)$ by implementing the Perceptron algorithm in the *primal form* in the feature space.

(e) Repeat **(b)** by using the hypothesis that you got in **(d)**. Please know that you need to map the points randomly generated in **(b)** by the nonlinear mapping ϕ first.

Ans: (a)

$$h(x) = Ax_1 + Bx_2 + C$$

where

$$A = -4.5, \quad B = 0, \quad C = 1$$

can be counted by following functions.

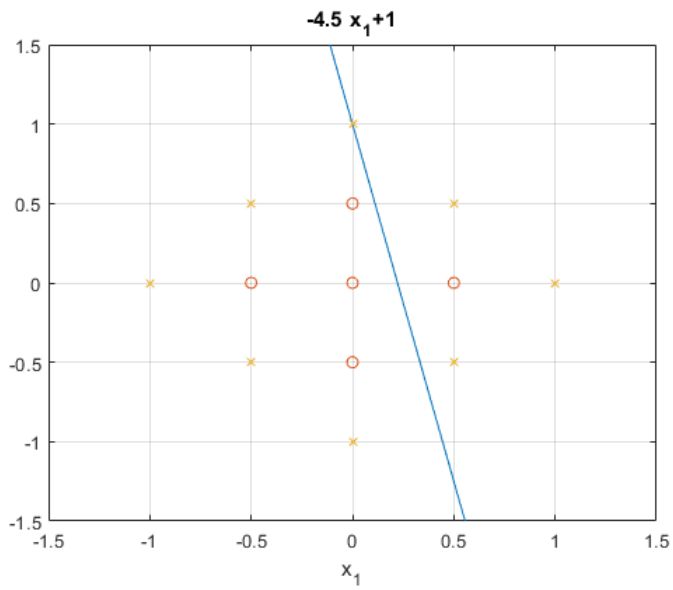
```
function plot_point(A_p, A_n)
%%%
% A_p = [0 0;0.5 0;0 0.5;-0.5 0;0 -0.5];
% A_n = [0.5 0.5;0.5 -0.5; -0.5 0.5;-0.5 -0.5;1 0;0 1;-1 0;0 -1];
% B_p = [0.5 0; 0 0.5; -0.5 0;0 -0.5];
% B_n = [0.5 0.5; 0.5 -0.5; -0.5 0.5; -0.5 -0.5];
%%%
    plot(A_p(:,1), A_p(:,2),'o');
    hold on
    plot(A_n(:,1), A_n(:,2),'x');
    m = 1.5;
```

```
    axis([-m m -m m]);
    axis square
    grid on
end
```

```
function [X, Y] = PNtoData(A_p, A_n)
    X = [A_p;A_n]';
    [m,¬] = size(A_p);
    [n,¬] = size(A_n);
    Y = [ones(m, 1);-ones(n, 1)];
end
```

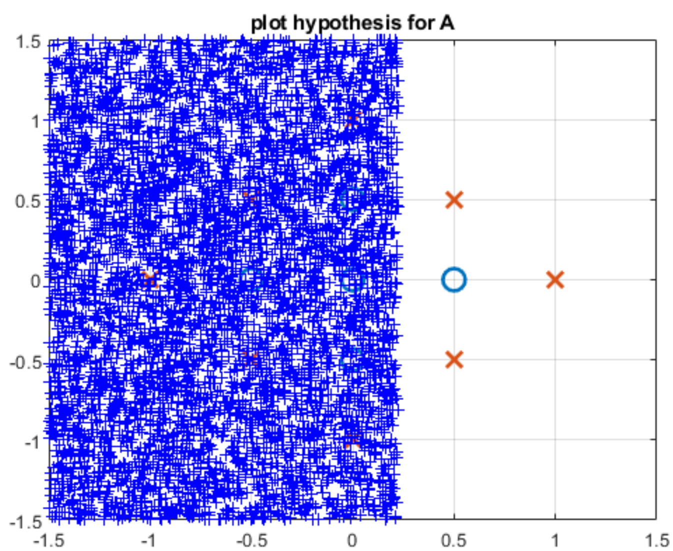
```
%%%dual form
function [a, b, L, check, times] = Perceptron2(X, Y)
% X(:,1), X(:,2),..., X(:, n) as column vectors
% m:
% n:
    [¬,n] = size(X);
    %%% count "L" (the max.)
    norms = [];
    for i = 1:n
        norms = [norms; norm(X(:,n), 2)];
    end
    L = max(norms);
    %%% count "L" (the max.)

    a = zeros(n, 1);
    b = 0;
    times = 0;
    tag = 1;

    while (tag) & (times < 1000)

        tag = 0;
        for i = 1:n

            %%% count "check" (the sigma)
            check  = 0;
            for j = 1:n
                check = check + a(j) * Y(i)* (Y(j)) * ((X(:,i)'*X(:,j))^2);
            end
            check = check + Y(i)*b;
            %%% count "check" (the sigma) end

            if check ≤ 0
                a(i) = a(i) + 1;
                b = b + Y(i)*L*L;
                tag = 1;
            end
        end
        times = times + 1;
    end
end
```

```
function [A, B, C] = DataToFunctionCoe(X, Y, a, b)
%%% A, B, C
% h(x1, x2) = A*x1 + B*x2 + C
    [m, n] = size(X)
    result = zeros(2,1);
    for i = 1:n
        result = result + eye(m) * a(i) * Y(i) * X(:,i);
    end
    A = result(1);
    B = result(2);
    C = b;
end
```



(b)

```
%for 4 (b) (c)
function plot_hypothesis2(X, Y)
%%%
% r = unifrnd(-1.5, 1.5, [2,10000]);
%%% A
% [X, Y] = PNtoData( [0 0;0.5 0;0 0.5;-0.5 0;0 -0.5],[0.5 0.5; 0.5 -0.5; ...
    -0.5 0.5;-0.5 -0.5;1 0;0 1;-1 0;0 -1])
%%% B
% [X, Y] = PNtoData( [0.5 0; 0 0.5; -0.5 0;0 -0.5], [0.5 0.5; 0.5 -0.5; ...
    -0.5 0.5; -0.5 -0.5])
%%%
    [a, b, L, check, times] = Perceptron2(X, Y);
    [A, B, C] = DataToFunctionCoe(X, Y, a, b);
    M = 10000;
    r = unifrnd(-1.5, 1.5, [2,M]);
    m = 1.5;
    for i = 1:M
        if (r(:,i)')*[A;B] + C > 0
            plot(r(1,i), r(2,i),'+b' );
            hold on;
        else
            %plot(r(1,i), r(2,i),'+r' );
        end
    end
    title('plot hypothesis') %for A, B
    axis([-m m -m m]);
end
```

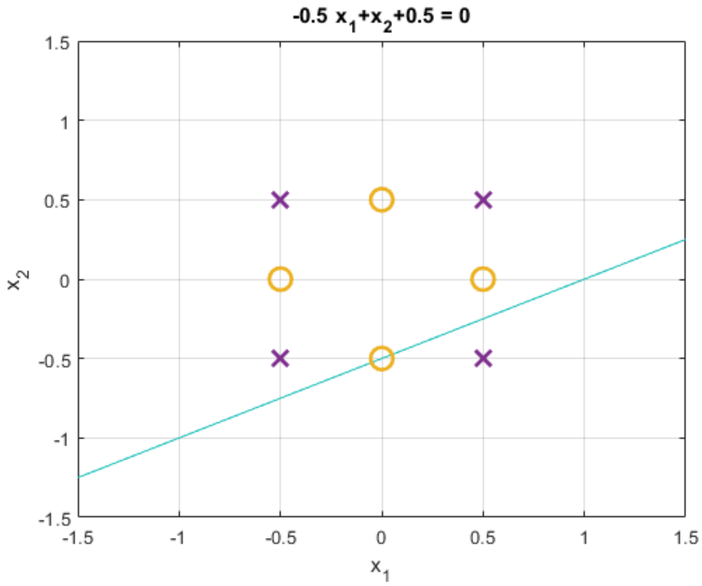


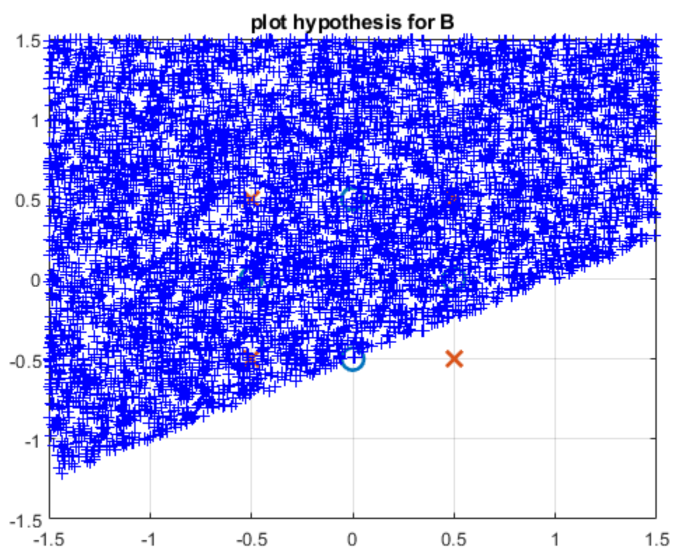
(c) hypothesis

$$h(x) = Ax_1 + Bx_2 + C$$

where

$$A = -0.5, \quad B = 1, \quad C = 0.5$$





(d)

$$f(x) = w^T \mathbf{x} + b$$

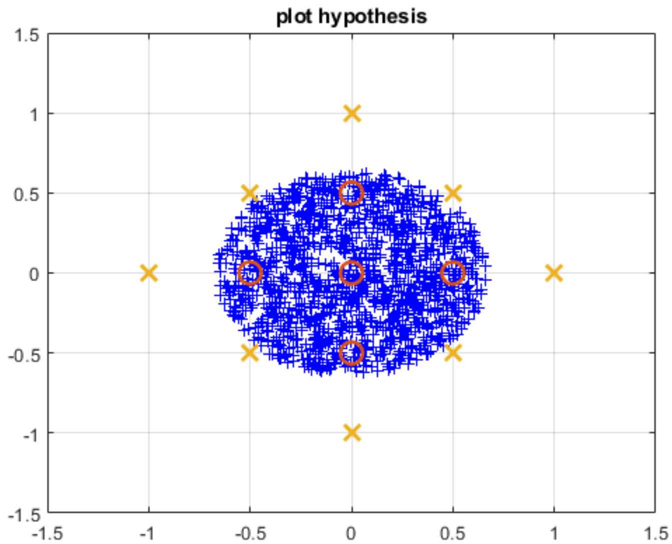
where

$$w = [\ w_1 \quad w_2 \quad w_3 \quad w_4\ ]^T = [\ 0 \quad -2.25 \quad 0 \quad -2.5\ ]^T, \quad b = 1$$

```
function Output = phi(X)
%%% ( R^2 )^n ---> ( R^4 ) ^n
% Mat_(2 x n) |---> Mat_(4 x n)
    Output1 = -X(1,:).*X(2,:);
    Output2 = X(1,:).^2;
    Output3 = X(1,:).*X(2,:);
    Output4 = X(2,:).^2;
    Output = [Output1;Output2;Output3;Output4]
end
```

```
%for 4(d)
%%%primal form
function [w, b, R, times] = Perceptron(X, Y)
% X = phi(X), Y = Y
% X(:,1), X(:,2),..., X(:, n) as column vectors
% m:
% n:
    [m,n] = size(X);
    %%% count "R" (the max.)
    norms = [];
    for i = 1:n
        norms = [norms; norm(X(:,n), 2)]; %1 or 2-norm?
    end
    R = max(norms);
    %%% count "R" (the max.)

    w = zeros(m, 1);
    b = 0;
    times = 0;
    tag = 1;

    while (tag) & (times < 10000)
        tag = 0;
        for i = 1:n
            if Y(i) * (w'*X(:,i) + b) ≤ 0
                w = w + Y(i)*X(:,i);
                b = b + Y(i)*R*R;
                tag = 1;
            end
        end
        times = times + 1;
    end
end
```

(e)




```
%for 4(e)
function plot_hypothesis4(X, Y)
%%%
% r = unifrnd(-1.5, 1.5, [2,10000]);
%%% A
% [X, Y] = PNtoData( [0 0;0.5 0;0 0.5;-0.5 0;0 -0.5],[0.5 0.5; 0.5 -0.5; ...
    -0.5 0.5;-0.5 -0.5;1 0;0 1;-1 0;0 -1])
%%%
    transformed_X = phi(X)
    [w, b, ¬, ¬] = Perceptron(transformed_X, Y)

    M = 10000;
    r = unifrnd(-1.5, 1.5, [2,M]);
    R = phi(r);
    m = 1.5;
    for i = 1:M
       if w'*R(:,i) +b > 0
           plot(r(1,i), r(2,i),'+b' );
           hold on;
       else
           %plot(r(1,i), r(2,i),'+r' );
       end
    end
    axis([-m m -m m]);
end
```