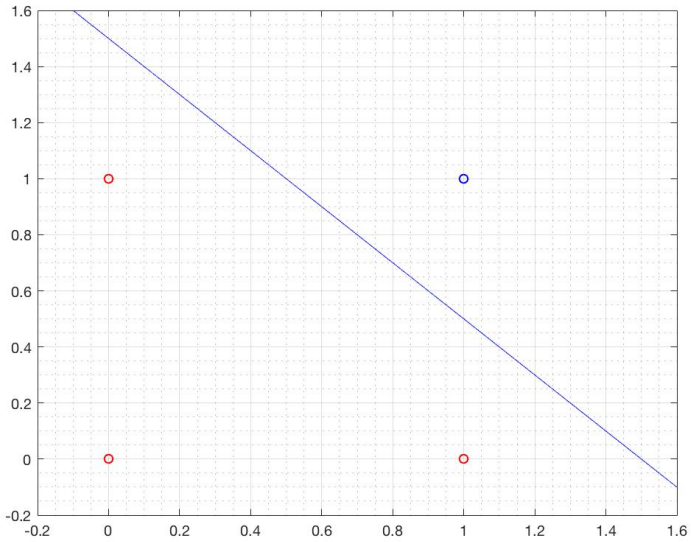
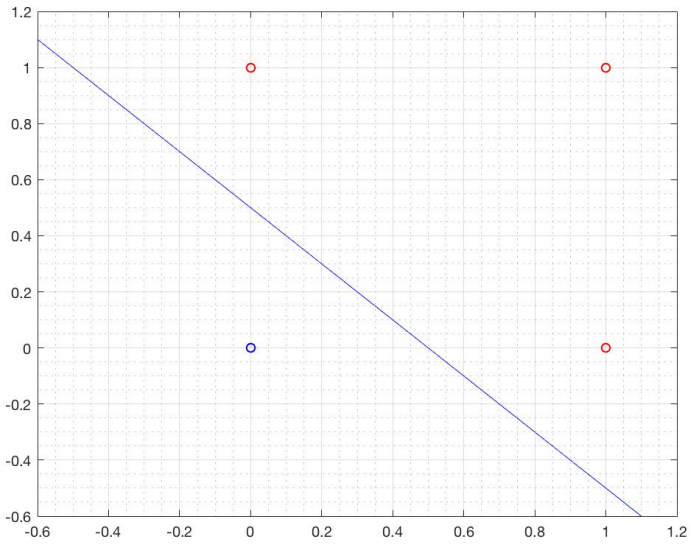
Q3

a) The implementation of the logic functions AND, OR, COMPLEMENT and NAND with selection of weights by off-line calculations are demonstrated as follows:

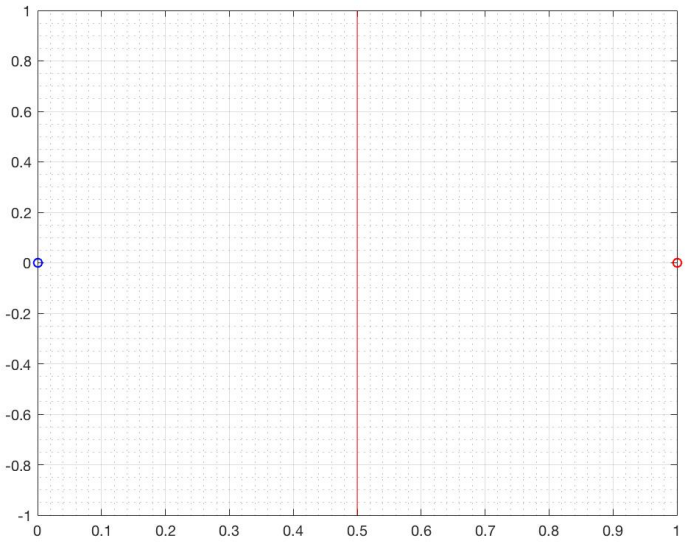
AND: $-1.5+x_1+x_2=0$, $w = [-1.5,1,1]^T$



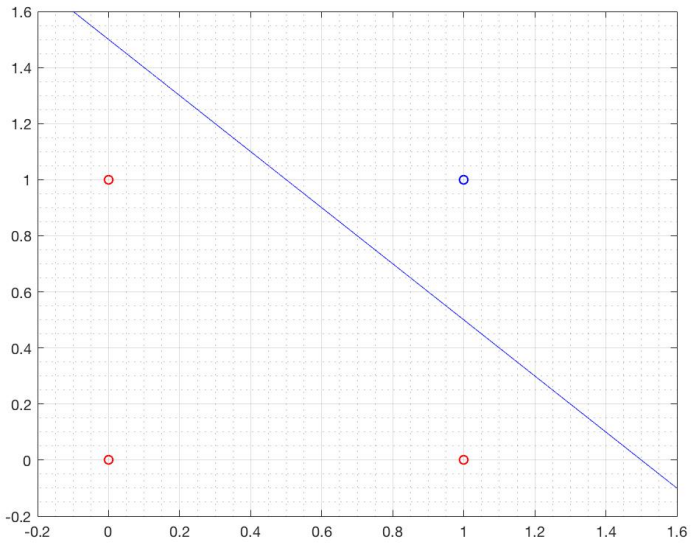
OR: $-0.5+x_1+x_2=0$, $w = [-0.5,1,1]^T$



COMPLEMENT: $0.5-x_1=0$, $w = [0.5,-1]^T$



NAND: $1.5-x_1-x_2=0$, $w = [1.5,-1,-1]^T$



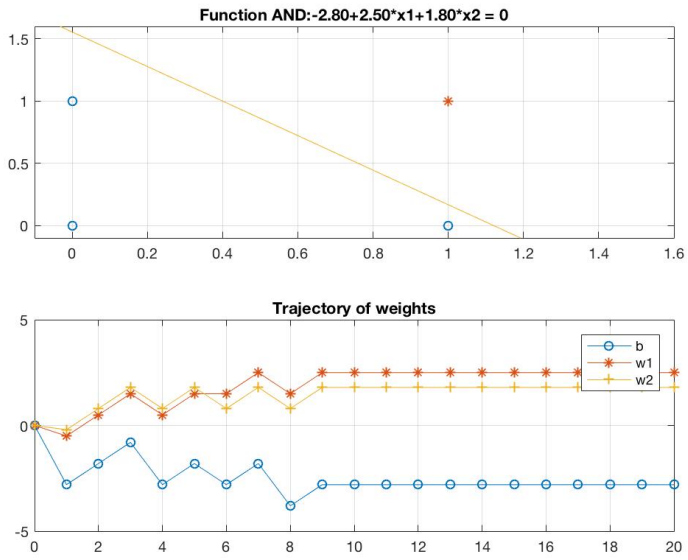
b) The main MATLAB code of learning procedure as follows:

```
n = 1;
P = [ones(1,length(d));P];
MAX = 20;
i = 0;
   while 1
   v = w * P;
```

```
y = hardlim(v);
e = (d-y);
ee(i+1) = mae(e);
    if (ee(i+1)<0.001)
    break;
    end
w = w+n*(d-y)*P';
i = i+1;
    if (i>=MAX)
    disp(w);
    break;
end
end
```

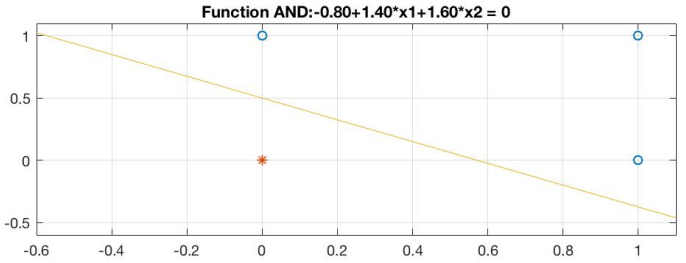
AND initialize:

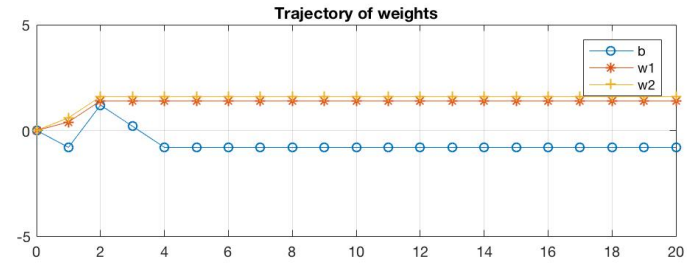
```
w = [0.2,0.5,0.8]; P = [0,0,1,1;0,1,0,1]; d = [0,0,0,1];
```



OR initialize:

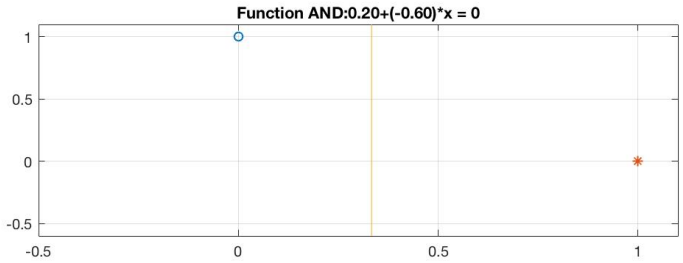
w = [0.2,0.4,0.6]; P = [0,0,1,1;0,1,0,1]; d = [0,1,1,1];

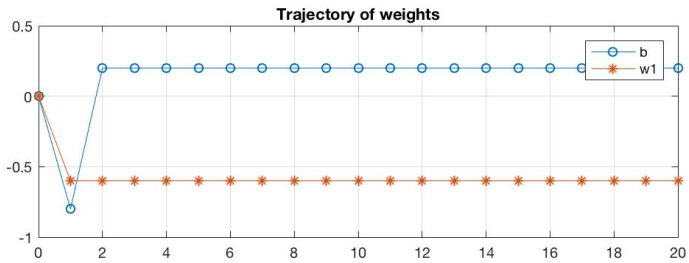




COMPLEMENT initialize:

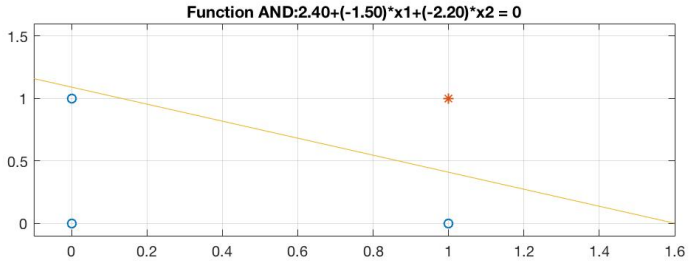
w = [0.2,0.4]; P = [0,1]; d = [1,0];

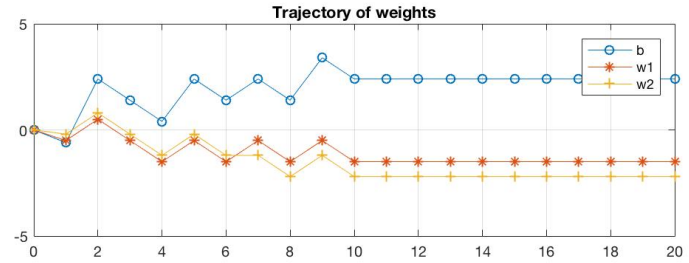




NAND initialize:

w = [0.4,0.5,0.8]; P = [0,0,1,1;0,1,0,1]; d = [1,1,1,0];





Comparing figures in (a) with figures in (b), although the specific weights vectors is different between off-line and learning procedure, all the decision boundaries could implementate the related logic functions correctly. When the learning rate is too small or too big, the result may not be accurate.

c) When apply the perception learning algorithm to the XOR function, the algorithm could not stop and the weight vector keeps waving. Because the XOR function are not linear separable and at least two straight lines are required to divide them, the algorithm could not converge in finite time and no correct decision boundary could be found.