Answer for Assignment 1

Dongmin Wu, 605308, Aalto University

Pen and perpaer exercise

Question 1:

No, if there is a matrix like this:

$$K = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

This matrix is a positive semidefinite matrix, but not all of its entries are positive.

Question 2:

Yes, according to the defination of Positive semi-definite matrix A, for any vector $v' \in \mathbb{R}^{n \times n}$, we have $v'Av \geq 0$.

So we can get

$$v'K_1v \geq 0, v'K_2v \geq 0$$

For K, we just need to verify if the $v'Kv \geq 0$ or not.

$$\begin{aligned} v'Kv &= v'(aK_1 + bK_2)v \\ &= av'K_1v + bv'K_2v \end{aligned}$$

Because $a, b \in \mathbb{R}^+$, we can derive

$$av'K_1v + bv'K_2v \geq 0$$

which means $v'Kv \geq 0$ is verified and K is a vaild kernel matrix.

Question 3

No, probably get negative result

Considering the situation below:

$$K_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, K_2 = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}$$

We can get

$$K = \begin{bmatrix} -2 & 0 \\ 0 & -2 \end{bmatrix}$$

Whose eigenvalue is negative, which is not a vaild kernel matrix

Question 4

$$
\begin{aligned}
h(x) &= sgn(||\phi(x)-c_-||^2 - ||\phi(x)-c_+||^2) \\
&= sgn(\phi(x)^2 - 2\phi(x)c_- + c_-^2 - \phi(x)^2 + 2\phi(x)c_+ - c_+^2) \\
&= sgn(-2\phi(x)c_- + c_-^2 + 2\phi(x)c_+ - 2c_+^2) \\
&= sgn(\frac{1}{2} \times (\frac{-1}{m_-}\Sigma_{i\in I^-}\langle\phi(x),\phi(x_i)\rangle + \frac{1}{2m_-^2}\Sigma_{i,j\in I^-}\langle\phi(x_i),\phi(x_j)\rangle + \frac{1}{m_+}\Sigma_{i\in I^+}\langle\phi(x),\phi(x_i)\rangle - \frac{1}{2m_+^2}\Sigma_{i,j\in I^+}\langle\phi(x_i),\phi(x_j)\rangle)) \\
&= sgn(\frac{1}{2} \times (\Sigma_{i\in I}\alpha_i\langle\phi(x),\phi(x_i)\rangle + \frac{1}{2m_-^2}\Sigma_{i,j\in I^-}\langle\phi(x_i),\phi(x_j)\rangle - \frac{1}{2m_+^2}\Sigma_{i,j\in I^+}\langle\phi(x_i),\phi(x_j)\rangle)) \\
&= sgn(\frac{1}{2} \times (\Sigma_{i\in I}\alpha_i\kappa(\phi(x),\phi(x_i)) + \frac{1}{2m_-^2}\kappa(\phi(x_i),\phi(x_j)) - \frac{1}{2m_+^2}\kappa(\phi(x_i),\phi(x_j)))) \\
&= sgn(\frac{1}{2} \times (\Sigma_{i\in I}\alpha_i\kappa(\phi(x),\phi(x_i)) + \frac{1}{2m_-^2}\kappa(\phi(x_i),\phi(x_j)) - \frac{1}{2m_+^2}\kappa(\phi(x_i),\phi(x_j)))) \\
&= sgn(\frac{1}{2} \times (\Sigma_{i\in I}\alpha_i\kappa(\phi(x),\phi(x_i)) + b))
\end{aligned}
$$

we can easily eminate the coefficient $\frac{1}{2}$ because of the property of sign function.

In conlution, we can get

$$h(x) = sgn(||\phi(x)-c_-||^2 - ||\phi(x)-c_+||^2) = sgn(\Sigma_{i\in I}\alpha_i\kappa(\phi(x),\phi(x_i)) + b)$$

Computer exercise

Question 5

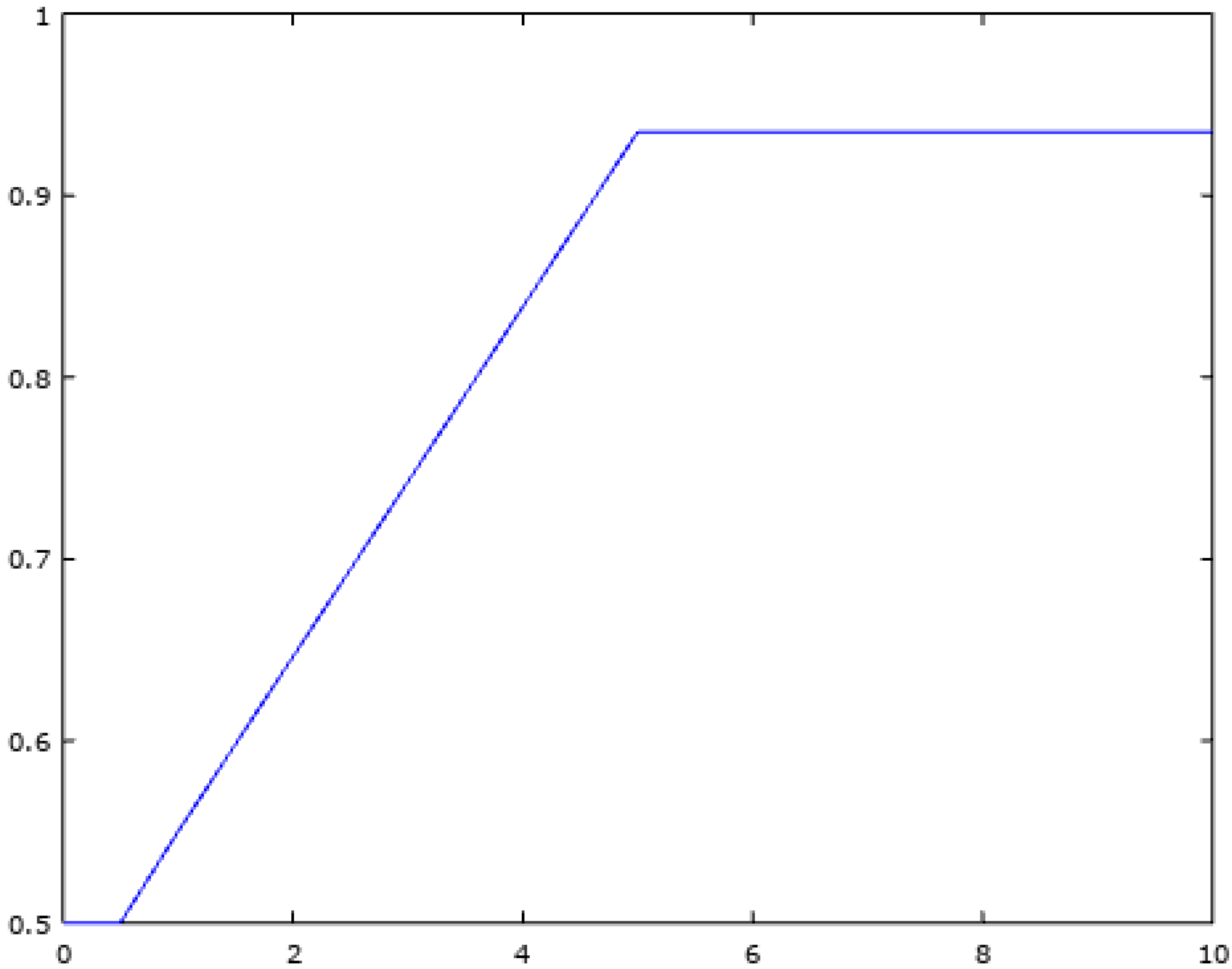
```
function ret = gaussianKernel(X, Z, S)

num_m = size(X,1);
num_z = size(Z,1);

myNorm = repmat(sum(X,2),1,num_z) - 2*X*Z' + repmat(sum(Z,2)',num_m,1);

ret = exp(- myNorm ./ (2 * (S^2) ));

end
```

Question 6

```
function ret = preditResult(X_train,X_test,labelX,S)
%myFun - Description
%
% Syntax: ret = myFun(input)
%
% Long description

num_X_train = size(X_train,1);
num_X_test = size(X_test,1);

X_train_pos = [];
X_train_neg = [];
X_test_pos = [];
X_test_neg = [];

for index = 1:num_X_train
    if labelX(index) >= 0
        X_train_pos = [X_train_pos;X_train(index,:)];
    else
        X_train_neg = [X_train_neg;X_train(index,:)];
    end
end



num_X_train_pos = size(X_train_pos,1);
num_X_train_neg = size(X_train_neg,1);



K_neg = gaussianKernel(X_train_neg,X_train_neg,S);
K_pos = gaussianKernel(X_train_pos,X_train_pos,S);

b = (1/(2*size(K_neg,1)^2))*sum(sum(K_neg,2)) - 1/(2*size(K_pos,1)^2)*sum(sum(K_pos,2));

ret_pos = 1/num_X_train_pos*(sum(gaussianKernel(X_train_pos,X_test,S)));
ret_neg = -1/num_X_train_neg*(sum(gaussianKernel(X_train_neg,X_test,S)));

hypothesis = ret_pos+ret_neg+b;

ret = [];
for i = 1:size(hypothesis,2)

    if hypothesis(:,i) >0
        ret = [ret;1];
    else
        ret = [ret;-1];
    end
end

end
```

Question 7

```
function ret = accuracyCalc(X_train, X_test, y_train,y_test,S)
    num_S = size(S,2);
    num_test = size(y_test,1);
    ret = [];

    for i = 1:num_S
        correct = 0;
        predit = preditResult(X_train,X_test,y_train,S(:,i));
        for j = 1:num_test
            if predit(j) == y_test(j)
                correct += 1;
            end
        end
        ret = [ret; 1.0*correct/num_test];
    end
end
```



Conclusion

Obviously, the result is not right, I suppose the reason is I have implemented a wrong $h(x)$. But I have checked the code few times and cannot find a solution for it, I think I may have some misunderstanding about the classfier funciton $h(x)$