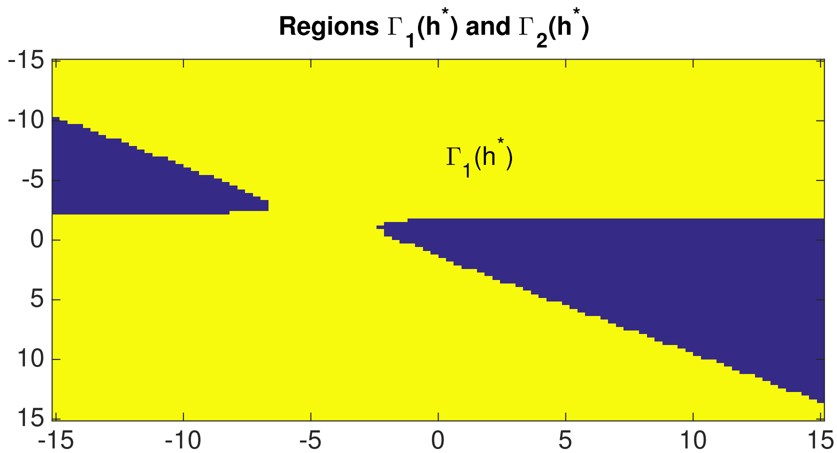

Figure 1:

Question 3

```
u1 = [0;1];
u2 = [1;0];

S1 = [3 -6; -6 18];
S2 = [1 0;0 1];

N = 100;
[X1,X2] = meshgrid(linspace(-15,15,N), linspace(-15,15,N));

X = [reshape(X1,N^2,1)';reshape(X2,N^2,1)'];

P1 = diag(1/2/pi/sqrt(det(S1)) * ... \\
exp(-0.5*(X - repmat(u1,1,N^2))'*inv(S1)*(X - repmat(u1,1,N^2))));
P2 = diag(1/2/pi/sqrt(det(S2)) * ...\\
exp(-0.5*(X - repmat(u2,1,N^2))'*inv(S2)*(X - repmat(u2,1,N^2))));

M1 = reshape((ones(size(P1)).*(P1 > P2))',N,N);

figure,
imagesc(linspace(-15,15,N),linspace(-15,15,N),(M1))
```

See figure **??**

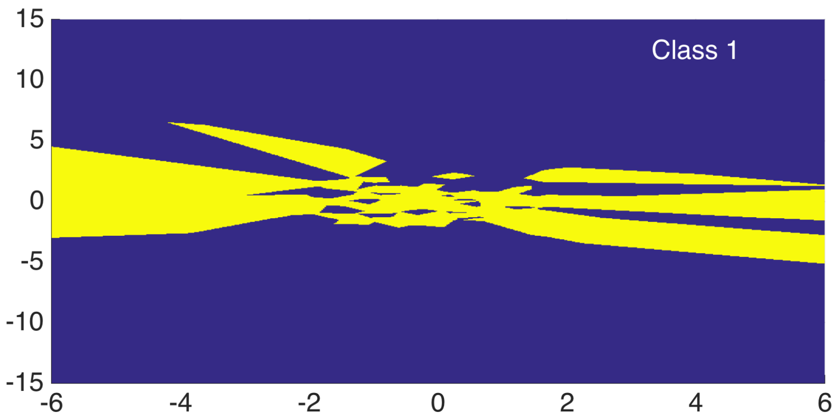

Figure 2:

Question 4

```
load hw10p4_data.mat

N = 500;
[Xtest1, Xtest2] = meshgrid(linspace(-6,6,N), linspace(-15,15,N));

X = [reshape(Xtest1,N^2,1) reshape(Xtest2,N^2,1)]';

Xtrain = [X1 X2];
C = zeros(1,size(X,2));

for ii = 1:size(X,2)
    dist = sum((repmat(X(:,ii),1,size(Xtrain,2)) - Xtrain).^2);
    [v,ind] = min(dist);
    C(ii) = (ind <= 100)*1 + (ind > 100)*2;
end

M = reshape(C',N,N);
figure,
surf(linspace(-6,6,N), linspace(15,-15,N),M)
```

See figure **??**

Question 5

```
N = 5000;

% points from class 1

u1 = [0; 1];
S1 = [3 -6; -6 18];

[V,D] = eig(S1);
Q1 = V*sqrt((D));
r1 = randn(2,N);
r1 = Q1*r1 + repmat(u1,1,size(r1,2));



% points from class 2

u2 = [1; 0];
S2 = [1 0; 0 1];

[V,D] = eig(S2);
Q2 = V*sqrt((D));
r2 = randn(2,N);
r2 = Q2*r2+repmat(u2,1,size(r1,2));

% Bayes generalization risk
r = [r1 r2];
C = zeros(1,2*N);

P1 = diag(1/2/pi/sqrt(det(S1)) * ... \\
 exp(-0.5*(r - repmat(u1,1,2*N))'*inv(S1)*(r - repmat(u1,1,2*N))));
P2 = diag(1/2/pi/sqrt(det(S2)) * ... \
exp(-0.5*(r - repmat(u2,1,2*N))'*inv(S2)*(r - repmat(u2,1,2*N))));

D = P1>=P2;

disp('Baesian risk:')
risk_Bayes = (sum(D(1:N) == 0) + sum(D(N+1:end) == 1))/2/N


% Nearest neighbor risk
load hw10p4_data.mat
Xtrain = [X1 X2];

C = zeros(1,size(r,2));

for ii = 1:size(r,2)
    dist = sum((repmat(r(:,ii),1,size(Xtrain,2)) - Xtrain).^2);
```

```
    [v,ind] = min(dist);
    C(ii) = (ind <= 100)*1 + (ind > 100)*2;
end

disp('NN risk:')
risk_NN = (sum(C(1:N) == 2) + sum(C(N+1:end) == 1))/2/N


Baesian risk:

risk_Bayes =

    0.1787

NN risk:

risk_NN =

    0.2892
```

Question 6

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Sat Dec  2 22:15:38 2017

@author: Kyle
"""


import numpy as np
import scipy as sp
import matplotlib.pyplot as plt


def estiGMM(Betas, Mus, Sigmas, X, tol=1e-3):
    K = len(Betas)
    M = Sigmas[0].shape[0]
    N = X.shape[0]
    log_phi = lambda mu, sigma, X: np.log(1/(2*np.pi)**(M/2.)/np.sqrt( \
                                   np.linalg.det(sigma))) - \
                                   (1./2)*np.sum(np.square((X-mu).dot( \
                                   sp.linalg.sqrtm(np.linalg.inv(sigma)))), \
                                   axis=1)

    Gammas = np.zeros([N, K])
    Log_Phis = np.zeros([N, K])
    for k in range(K):
        Log_Phis[:,k] = log_phi(Mus[k], Sigmas[k], X)
        Gammas[:,k] = Betas[k] * np.exp(Log_Phis[:,k])
    temp = np.sum(Gammas, axis=1)
    Gammas = Gammas / temp[:,None]
    ll = 0;
    for k in range(K):
        ll += np.sum(Gammas[:,k] * (np.log(Betas[k]) + Log_Phis[:,k]))
    ll_pre = 2 * ll
#    i = 0

    while (ll - ll_pre) > tol:
#        i += 1
        ll_pre = ll
        for k in range(K):
            gamma = Gammas[:,k]
            temp = np.sum(gamma)
            Betas[k] = 1./N * temp
            Mus[k] = np.sum(gamma[:,None]*X, axis=0)/temp
            Sigmas[k] = (X-Mus[k]).transpose().dot( \
                        gamma[:,None] * (X-Mus[k])) \
```

```
                             / temp
        for k in range(K):
            Log_Phis[:,k] = log_phi(Mus[k], Sigmas[k], X)
            Gammas[:,k] = Betas[k] * np.exp(Log_Phis[:,k])
        temp = np.sum(Gammas, axis=1)
        Gammas = Gammas / temp[:,None]
        ll = 0
        for k in range(K):
            ll += np.sum(Gammas[:,k] * (np.log(Betas[k]) + Log_Phis[:,k]))
#         print(ll)
#
#     print(i)
    return {'Betas': Betas, 'Mus': Mus, 'Sigmas': Sigmas}


if __name__ == '__main__':
    Data_Train = sp.io.loadmat('hw10p6_data.mat')
    X  = Data_Train['X'].transpose()
    plt.figure()
    plt.scatter(X[:,0], X[:,1])

    K = 5
    Betas = [1./5] * K
    mu1 = np.array([-.9, .1])
    mu2 = np.array([-.5, .06])
    mu3 = np.array([0., 0.])
    mu4 = np.array([.5, 0.])
    mu5 = np.array([.8, -.1])
#     mu1 = np.array([-.9, .1])
#     mu2 = np.array([-.7, .08])
#     mu3 = np.array([-.25, .03])
#     mu4 = np.array([.25, 0.])
#     mu5 = np.array([.75, -.1])
    Mus = [mu1, mu2, mu3, mu4, mu5]
    Sigmas = [np.identity(2) * .1] * K
    Theta = estiGMM(Betas, Mus, Sigmas, X, tol=1e-3)
    Betas = Theta['Betas']
    Mus = Theta['Mus']
    Sigmas = Theta['Sigmas']



    N = 200
    lx1 = -1.2; ux1 = 1.2
    lx2 = -.16; ux2 = .26
    x1 = np.linspace(lx1, ux1, N)
    x2 = np.linspace(lx2, ux2, N)
    X1, X2 = np.meshgrid(x1, x2)
```

```
    l = np.zeros([N, N])
    for i in range(N):
        for j in range(N):
            x = np.array([X1[i,j], X2[i,j]])
            for k in range(K):
                l[i,j] += Betas[k]/(2*np.pi)/np.sqrt(np.linalg.det \
                          (Sigmas[k])) * np.exp((-1./2)*(x-Mus[k]).dot( \
                          np.linalg.inv(Sigmas[k]).dot(x-Mus[k])))
    plt.contour(X1, X2, l)

    plt.figure()
    plt.scatter(X[:,0], X[:,1])
    XX = np.zeros([N**2, 2])
    XX[:,0] = np.reshape(X1, N**2)
    XX[:,1] = np.reshape(X2, N**2)
    for k in range(K):
        det_sigma = np.linalg.det(Sigmas[k])
        sqrtm_inv_sigma = sp.linalg.sqrtm(np.linalg.inv(Sigmas[k]))
        gl = np.exp(np.log((2*np.pi)/np.sqrt(det_sigma)) - \
                (1./2)*np.sum(np.square((XX-Mus[k]).dot( \
                sqrtm_inv_sigma)), axis=1))
        plt.contour(X1, X2, np.reshape(gl, [N,N]))
```

Estimated parameters:

```
Beta: [0.0908958335806793 06,
 0.23796749364501341,
 0.34897510833542866,
 0.08624417064824 3773,
 0.23591739379063484]

 Mu:

 [array([-0.88912192,  0.12334891]),
 array([-0.54224958,  0.08728456]),
 array([ 0.03942024,  0.00931536]),
 array([ 0.90364802, -0.07316714]),
 array([ 0.57621427, -0.03048865])]

 Sigmas[0]
 array([[ 0.00370032,  0.00267617],
        [ 0.00267617,  0.00272899]])

Sigmas[1]
array([[ 0.02605784, -0.00989869],
       [-0.00989869,  0.00427219]])

Sigmas[2]
array([[ 0.04261283,  0.00341552],
```

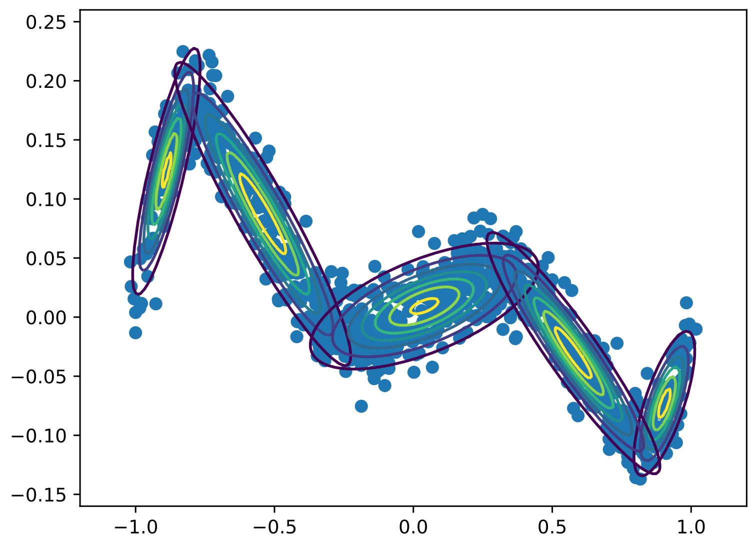

Figure 3:

```
        [ 0.00341552,  0.00071867]])

Sigmas[3]
        array([[ 0.00319583,  0.00131645],
        [ 0.00131645,  0.00100055]])

 Sigmas[4]
 array([[ 0.02381501, -0.00731792],
        [-0.00731792,  0.00256006]])
```

See figure **??**