In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import mltools as ml
from logisticClassify2 import *
```

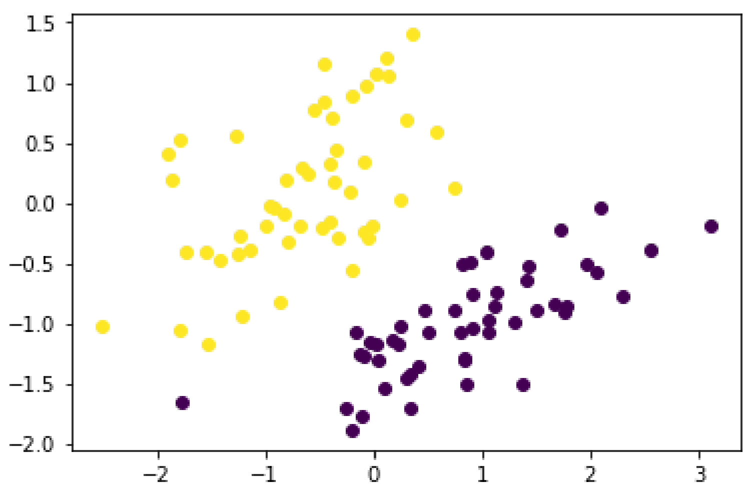
# Problem 1

In [2]:
```
iris = np.genfromtxt("data/iris.txt",delimiter=None)
X, Y = iris[:,0:2], iris[:,-1] # get first two features & target
X,Y = ml.shuffleData(X,Y) # order randomly rather than by class label
X,_ = ml.transforms.rescale(X) # rescale to improve numerical stability, speed convergence
XA, YA = X[Y<2,:], Y[Y<2] # Dataset A: class 0 vs class 1
XB, YB = X[Y>0,:], Y[Y>0] # Dataset B: class 1 vs class 2
```

## Problem 1.1

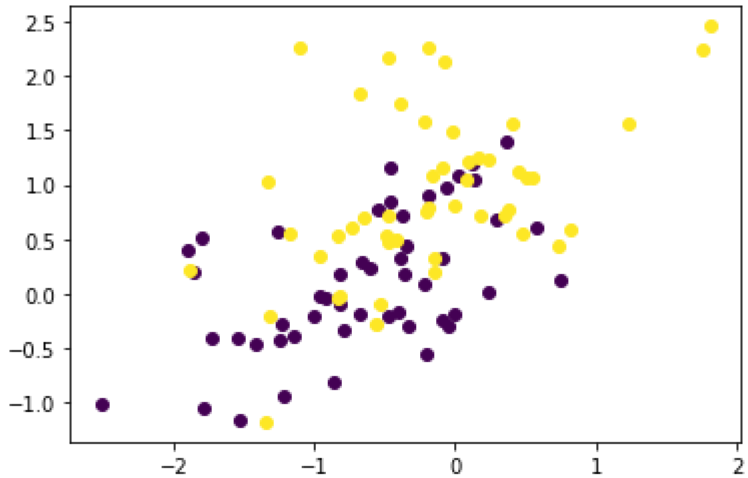
In [3]:
```
print("Class 0 vs Class 1")
ml.plotClassify2D(None,XA,YA)
```

```
Class 0 vs Class 1
```



In [4]:
```
print("Class 1 vs Class 2")
ml.plotClassify2D(None,XB,YB)
```

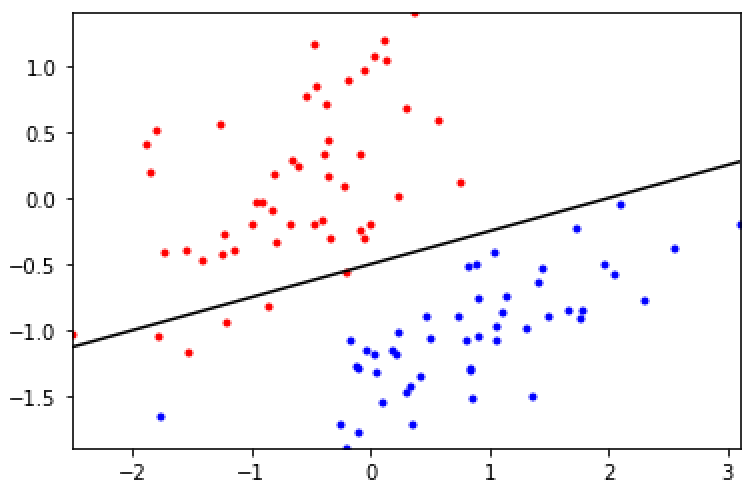
```
Class 1 vs Class 2
```



## Problem 1.2

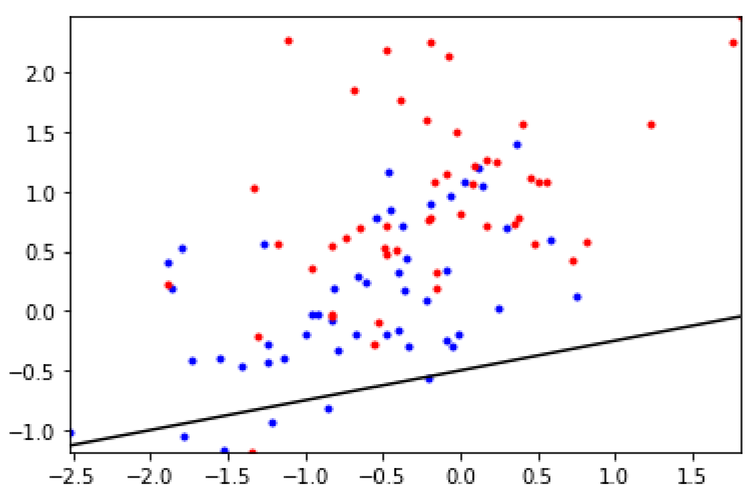
In [5]:
```
print("Prediction on Class 0 and Class 1 by funtion plotBoundary")
learnerA = logisticClassify2()
learnerA.classes = np.unique(YA)
wts = np.array([0.5,-0.25,1])
learnerA.theta = wts
learnerA.plotBoundary(XA,YA)
```

```
Prediction on Class 0 and Class 1 by funtion plotBoundary
```



In [6]:
```
print("Prediction on Class 1 and Class 2 by function plotBoundary")
learnerB = logisticClassify2()
learnerB.classes = np.unique(YB)
wts = np.array([0.5,-0.25,1])
learnerB.theta = wts
learnerB.plotBoundary(XB,YB)
```

```
Prediction on Class 1 and Class 2 by function plotBoundary
```



In [ ]:
```
"""
def plotBoundary(self,X,Y):
    if len(self.theta) != 3: raise ValueError('Data & model must be 2D');
    ax = X.min(0),X.max(0); ax = (ax[0][0],ax[1][0],ax[0][1],ax[1][1]);
    x1b = np.array([ax[0],ax[1]]);

    ### Lines added to the plotBoundary function
    x2bx = (-self.theta[1]*x1b[0] - self.theta[0])/self.theta[2]
    x2by = (-self.theta[1]*x1b[1] - self.theta[0])/self.theta[2]
    ### Lines added to the plotBoundary function

    x2b = np.array([x2bx,x2by])
    A = Y==self.classes[0];
    plt.plot(X[A,0],X[A,1],'b.',X[~A,0],X[~A,1],'r.',x1b,x2b,'k-'); plt.axis(ax); plt.draw
"""
```

## Problem 1.3

In [8]:
```
print("Learner A's error rate: ", learnerA.err(XA,YA))
print("Learner B's error rate: ", learnerB.err(XB,YB))
```

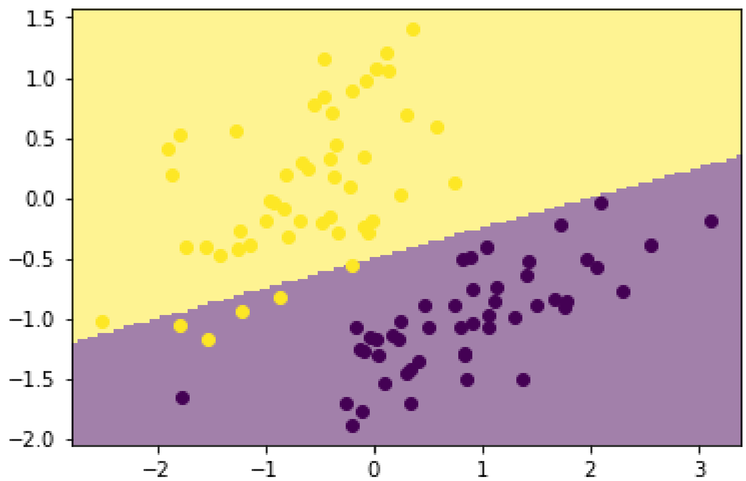
```
Learner A's error rate:  0.050505050505050504
Learner B's error rate:  0.46464646464646464
```

In [ ]:
```
"""
def predict(self, X):

    ### Lines added to the plotBoundary function
    r = []
    Yhat = np.array([])
    for i in enumerate(X):
        r.append(self.theta[0] + self.theta[1]*X[i[0],0] + self.theta[2]*X[i[0],1])
        if r[i[0]] > 0:
            Yhat = np.append(Yhat,self.classes[1])
        else:
            Yhat = np.append(Yhat,self.classes[0])
    ### Lines added to the plotBoundary function

    return Yhat
"""
```

## Problem 1.4

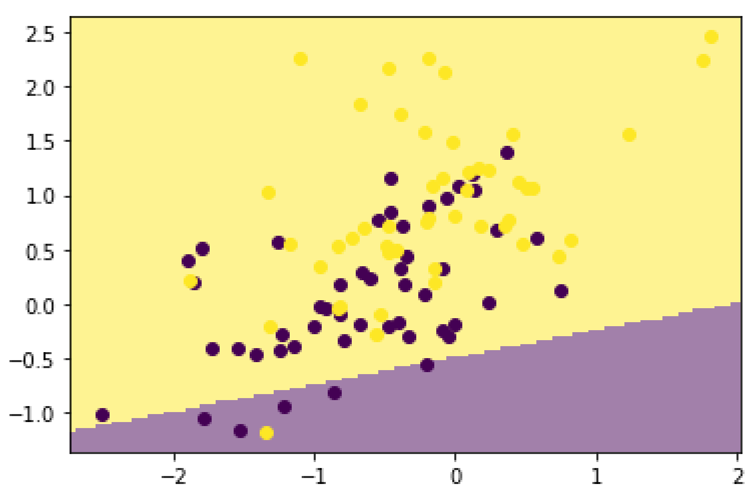
In [10]:
```
print("Prediction on Class 0 and Class 1 by funtion plotClassify2D")
ml.plotClassify2D(learnerA,XA,YA)
```

Prediction on Class 0 and Class 1 by funtion plotClassify2D



In [11]:
```
print("Prediction on Class 0 and Class 1 by funtion plotClassify2D")
ml.plotClassify2D(learnerB,XB,YB)
```

Prediction on Class 0 and Class 1 by funtion plotClassify2D



## Problem 1.5

$$\nabla_\theta J(\theta) = (\sigma(\theta x^i) - y^i)\ \sigma(\theta x^i)\ x^i$$
$$J_\theta(\theta) = -y^{(j)} log\sigma(x^{(j)} * \theta) - (1 - y^{(j)}) log(1 - \sigma(x^{(j)} * \theta))$$

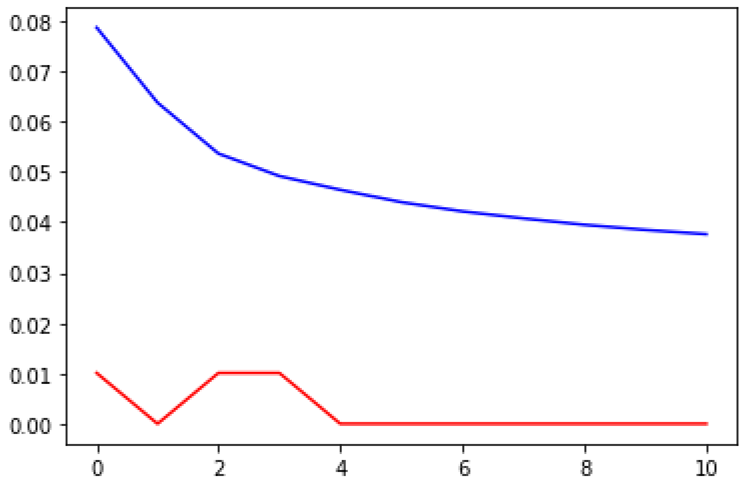
## Problem 1.6

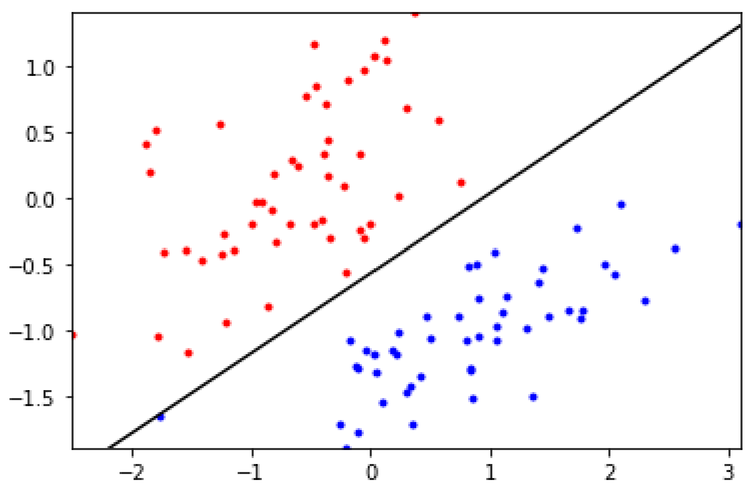
In [ ]:
```
"""
    def sigmoid(self,x):
        return 1./ (1+np.exp(-x))

    def train(self, X, Y, initStep=1.0, stopTol=1e-4, stopEpochs=5000, plot=None):
        M,N = X.shape;                     # initialize the model if necessary:
        self.classes = np.unique(Y);       # Y may have two classes, any values
        XX = np.hstack((np.ones((M,1)),X)) # XX is X, but with an extra column of ones
        YY = ml.toIndex(Y,self.classes);   # YY is Y, but with canonical values 0 or 1
        if len(self.theta)!=N+1: self.theta=np.random.rand(N+1);
        epoch=0; done=False; Jnll=[]; J01=[];
        while not done:
            stepsize, epoch = initStep*2.0/(2.0+epoch), epoch+1; # update stepsize
            for i in np.random.permutation(M):
                ri = np.dot(XX[i],self.theta);     # TODO: compute linear response r(x)
                gradi = XX[i]*(self.sigmoid(ri)-YY[i])
                self.theta -= stepsize * gradi;  # take a gradient step
            J01.append(self.err(X,Y))

            Jsur = 0
            for i in range(M):
                if YY[i]==1:
                    Jsur+= -np.log(self.sigmoid(self.theta.dot(XX[i])))
                else:
                    Jsur+= -np.log(1-self.sigmoid(self.theta.dot(XX[i])))

            Jnll.append( Jsur/M )
            if epoch > stopEpochs or (len(Jnll)>1 and abs(Jnll[-2] - Jnll[-1]) < stopTol):
                done = True
        plt.figure(1); plt.plot(Jnll,'b-',J01,'r-'); plt.draw();    # plot losses
        if N==2: plt.figure(2); self.plotBoundary(X,Y); plt.draw(); # & predictor if 2D
"""
```

## Problem 1.7

In [13]:
```
print("Dataset XA and YA")
learnerA2 = logisticClassify2()
#learnerA2.classes = np.unique(YA)
#learnerA2.theta = np.array([0.5,-0.25, 1.])
learnerA2.train(XA,YA,initStep=0.5,stopTol=1e-3)
```

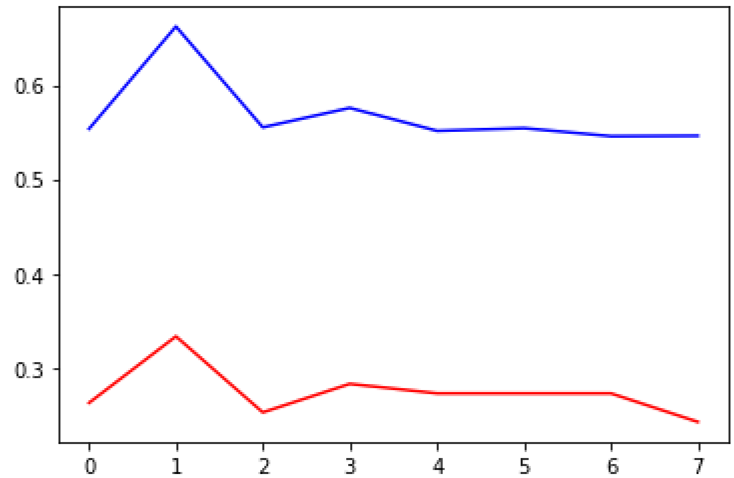
```
Dataset XA and YA
```

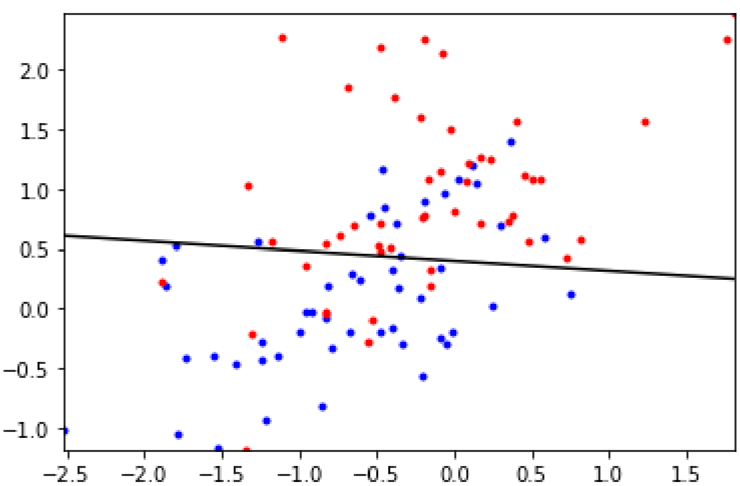


In [18]:
```
print("Dataset XB and YB")
learnerB2 = logisticClassify2()
learnerB2.train(XB,YB,0.5,1e-3);
```

```
Dataset XB and YB
```



# Problem 2

### Problem 2.1

vector x = [x1, x1*x2]

After mapping 4 points, we get ((x1,x1x2),y): ((1,1),1) ((-1,1),1) ((-1,-1),-1) ((1,-1),-1)

The positive examples have x1x2 = − 1 and the negative examples have x1x2 = + 1

Thus, the line is x1x2 = 0 and the vector w is [0,1]

Corresponding margin is 2

### Problem 2.2 will be shown at the end of the pdf.

### Problem 2.3

(1,1) and (1,0) are two points closest to the slope.

In order to make these two points same distance to the slope, the slope must go through (1, 0.5)

In order for one slope to go through (1,0) (0,1) at the same time, it should have slope = -1

Thus, put (1,0.5) into x2 = -x1 + b

We get x1 + x2 = 3/2, and the weight vector is (1, 1)

### Problem 2.4

The optimal margin increases when we remove (1, 0) or (1, 1) and stays the same when we remove others.

The reason is (1, 0) and (1, 1) are the closest two points to the line.

If we remove them, the margin will increases to the next closest points.

# Problem 3

I have followed the academic honesty guidelines posted on the course website

Problem 2.2