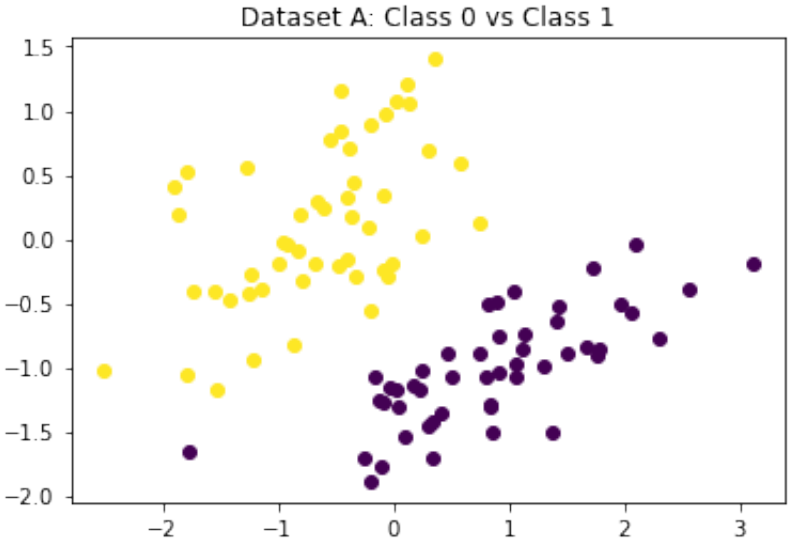
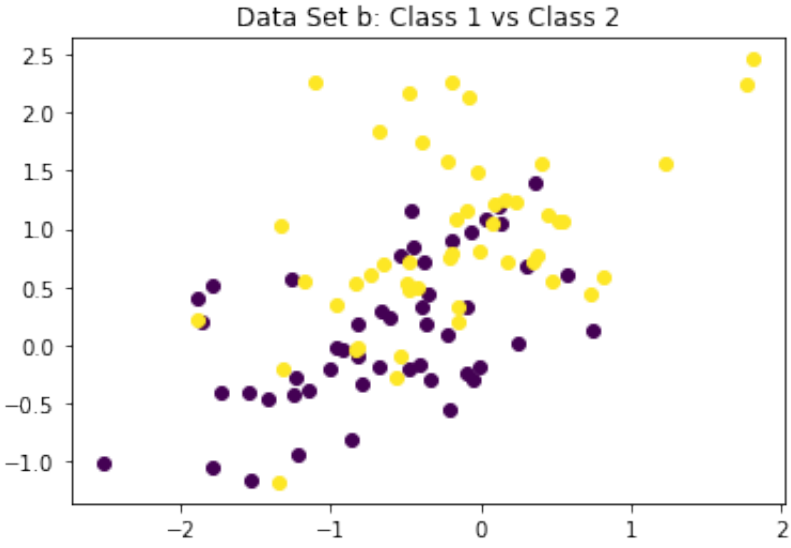
In [23]:
```
import numpy as np
import mltools as ml
np.random.seed(0)
import matplotlib.pyplot as plt
```

Problem 1.1

In [24]:
```
iris = np.genfromtxt("data/iris.txt",delimiter=None)
X, Y = iris[:,0:2], iris[:,-1]
X,Y = ml.shuffleData(X,Y)
X,_ = ml.transforms.rescale(X)
XA, YA = X[Y<2,:], Y[Y<2]
XB, YB = X[Y>0,:], Y[Y>0]
```

In [25]:
```
plt.title("Dataset A: Class 0 vs Class 1")
ml.plotClassify2D(None,XA,YA)
plt.show()
plt.title("Data Set b: Class 1 vs Class 2")
ml.plotClassify2D(None,XB,YB)
plt.show()
```

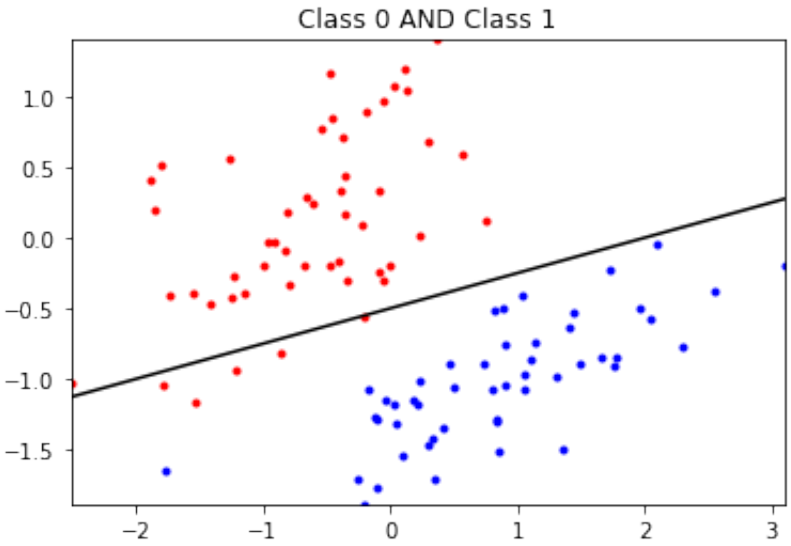




set A is linearly separable set B is not linearly separable.

Problem 1.2

In [26]:
```
from logisticClassify2 import *
learnerA = logisticClassify2(); # create "blank" learner
learnerA.classes = np.unique(YA) # define class labels using YA or 
wts = np.array([0.5, -0.25, 1.]); # TODO: fill in values
learnerA.theta = wts;

plt.title("Class 0 AND Class 1")
learnerA.plotBoundary(XA,YA)
plt.show()
```

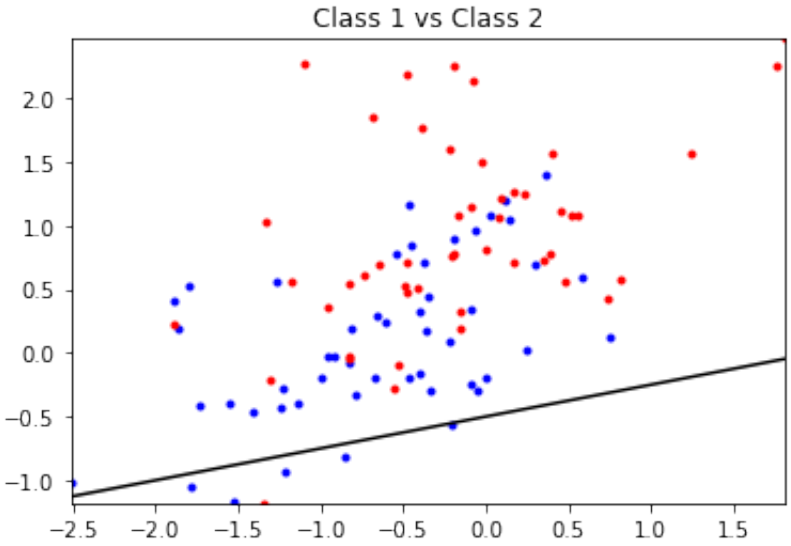


In [ ]:
```

```

My code

```
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]]);  # The X1 coordinates of the 
two points
    x2b = (-self.theta[0]-self.theta[1]*x1b)/self.theta[2];
    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();
```

In [27]:
```
learnerB = logisticClassify2();
learnerB.classes = np.unique(YB)
wts = np.array([0.5, -0.25, 1.])
learnerB.theta = wts

plt.title("Class 1 vs Class 2")
learnerB.plotBoundary(XB,YB)
plt.show()
```



Problem 1.3

In [28]:
```
learnerA = logisticClassify2()
learnerA.classes = np.unique(YA)
wts = np.array( [0.5,-0.25,1] )
learnerA.theta = wts
print("error rates is " + str(learnerA.err(XA, YA)))
```

```
error rates is 0.050505050505050504
```

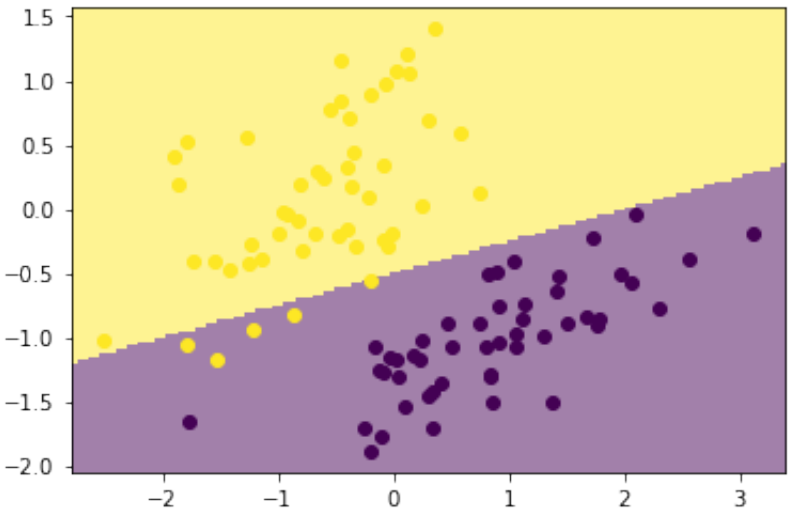
```
My code def predict(self, X):
    P = self.theta[0] + X.dot(self.theta[1:])
    Y01 = (P > 0).astype(int)
    Yhat = np.asarray(self.classes)[Y01]
    return Yhat
```

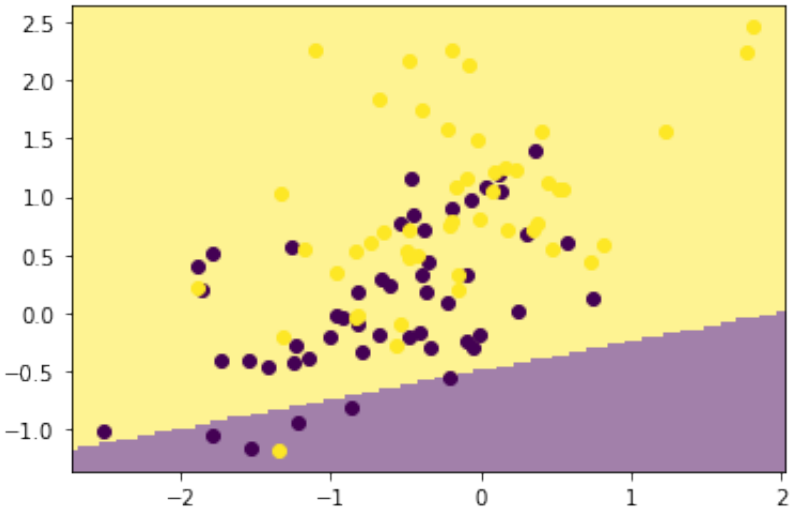
In [29]:
```
learnerB = logisticClassify2()
learnerB.classes = np.unique(YB)
wts = np.array( [0.5,-0.25,1] )
learnerB.theta = wts
print("error rates is " + str(learnerB.err(XB, YB)))
```

```
error rates is 0.46464646464646464
```

Problem 1.4

In [30]:
```
ml.plotClassify2D(learnerA,XA,YA)
plt.show()
ml.plotClassify2D(learnerB,XB,YB)
plt.show()
```





Problem 1.5

```
negative log-likelihood loss is
Jj(θ) = - log(σ(x^(j)· θ)) if y^(j) = 1
Jj(θ) = - log(1 - σ(x^(j)· θ)) if y^(j) = 0
```

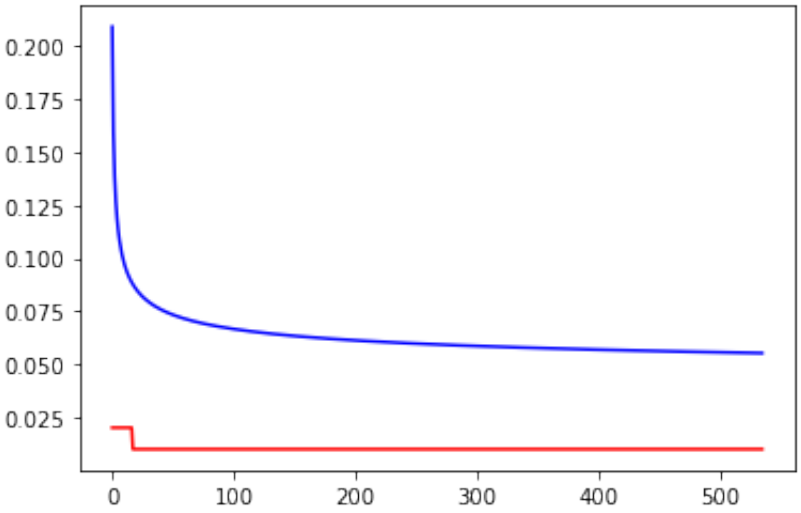
```
The gradient is:
∇Jj(θ) = - (1 - σ(x^(j)· θ)) x^(j) if y^(j) = 1
∇Jj(θ) = -σ(x^(j)· θ) x^(j) if y^(j) = 0
```

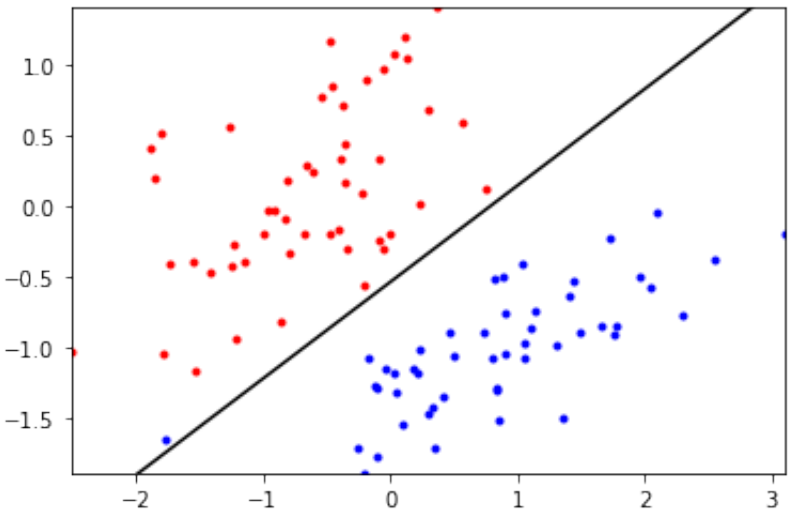
```
My train function:

def train(self,X,Y, initStep=1.,stopTol=1e-
4,stopEpochs=5000,plot=None):
    M,N = X.shape;
    self.classes = np.unique(Y);
    XX = np.hstack((np.ones((M,1)),X))
    YY = ml.toIndex(Y,self.classes);
    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;

        for i in np.random.permutation(M):
            ri = XX[i].dot(self.theta)
            si = 1./(1.+np.exp(-ri))
            gradi = -(1-si)*XX[i,:] if YY[i] else si*XX[i,:]
            self.theta -= stepsize * gradi;
        J01.append( self.err(X,Y) )

        S = 1./(1.+np.exp(-(XX.dot(self.theta))))
        Jsur = -np.mean(YY*np.log(S)+(1-YY)*np.log(1-S))
        Jnll.append( Jsur )
        plt.pause(.01);

        done = epoch>=stopEpochs or (epoch>1 and abs(Jnll[-1]-
Jnll[-2])<stopTol)


    plt.figure(1); plt.clf(); plt.plot(Jnll,'b-',J01,'r-');
plt.draw();
    if N==2: plt.figure(2); plt.clf(); self.plotBoundary(X,Y);
plt.draw();
```

Problem 1.6

In [31]:
```
learnerA = logisticClassify2()
learnerA.theta = np.array([0.,0.,0.]);
learnerA.train(XA,YA,initStep=1e-1,stopEpochs=1000,stopTol=1e-5);
print(learnerA.err(XA,YA))
plt.show()
```

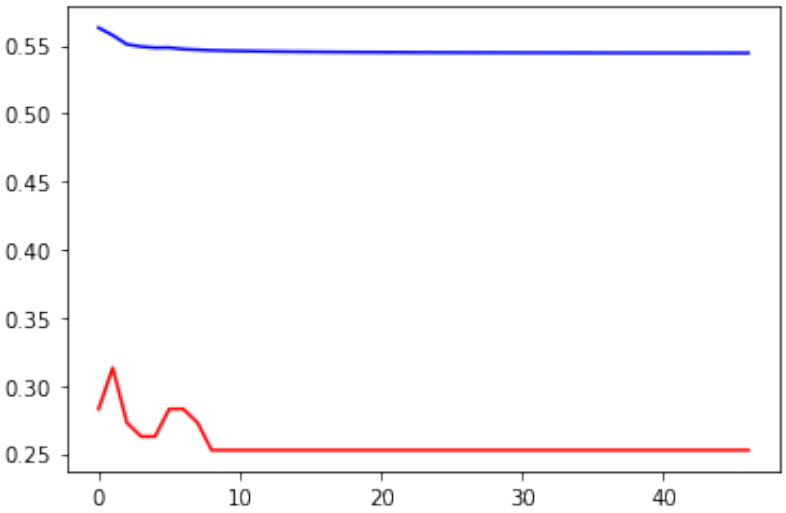
```
0.010101010101010102
```

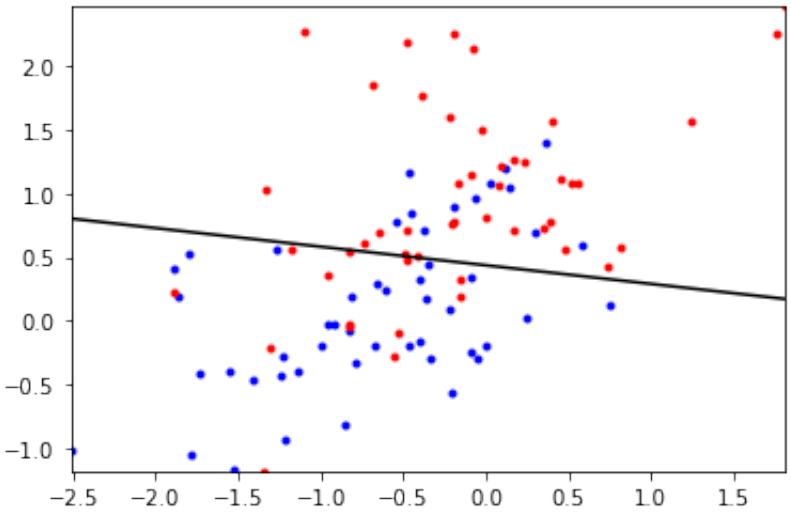




In [32]:
```
learnerB = logisticClassify2()
learnerB.theta = np.array([0.,0.,0.])
learnerB.train(XB,YB,initStep=1e-1,stopEpochs=1000,stopTol=1e-5)
print("training error rate: " + str(learnerB.err(XB,YB)))
plt.show()
```

```
training error rate: 0.25252525252525254
```





Problem2.1

T(a+bx1)is a vertical or horizontal linear classifier which is a line. This learner can shatter dataset (a) and (b) but not (c) or (d) because (a) and (b) can be separated. While (c) and (d), there are different points on the same side.

Problem 2.2

T((a * b)x1 + (c/a)x2) is a linear classifier passes through the origin. It is same as 2.1 that it can shatter data points in (a) and (b) but not shatter the data points in (c) or (d) because (a) and (b) can be separated. While (c) and (d), there are different points on the same side .

Problem 2.3

T((x1−a)^2+(x2−b)^2+c), is a random ciclre classifer. It can shatter data points in (a), (b) and (c) We can separate points in (a), Because in (d) dataset, it may have different values on the same side.

Problem 2.4 ec

T(a+bx1+cx2)×T(d+bx1+cx2) is parrelal linear classifier Learner C can shatter data points in (a), (b) and (c), but not shatter the data points in (d). The classifier creates three regions.

Problem 3

I did this homework independently.

In [ ]: