Name: Hootan Hosseinzadeganbushehri

[redacted]

UCI Email: hootanh@uci.edu

In [94]:
```python
from __future__ import division
import numpy as np
np.random.seed(0)
import mltools as ml
import sys
sys.path.append('code')
import matplotlib.pyplot as plt
plt.set_cmap('jet');
%matplotlib inline
import warnings
warnings.filterwarnings('ignore');
```

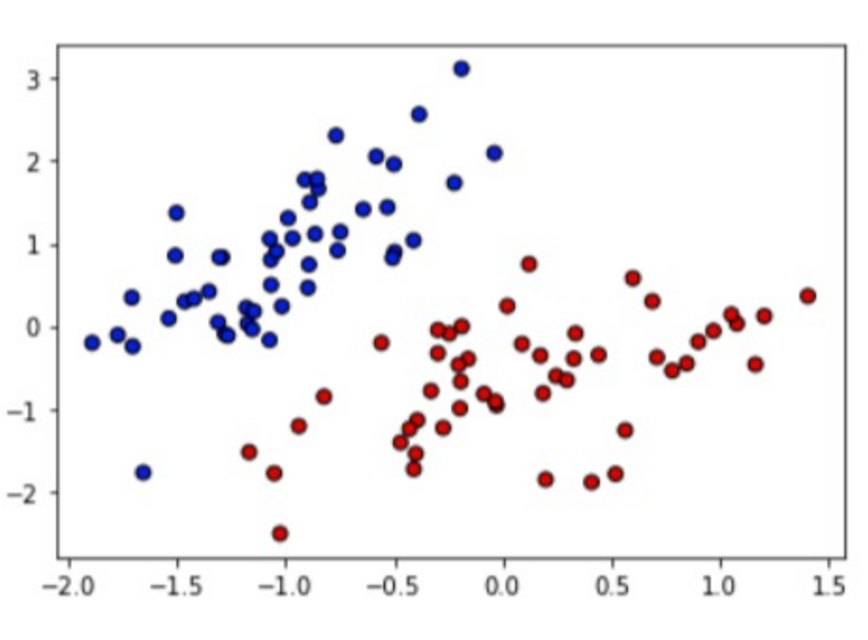
Problem 1:

In [95]:
```python
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]
```

Problem 1 - Part 1:

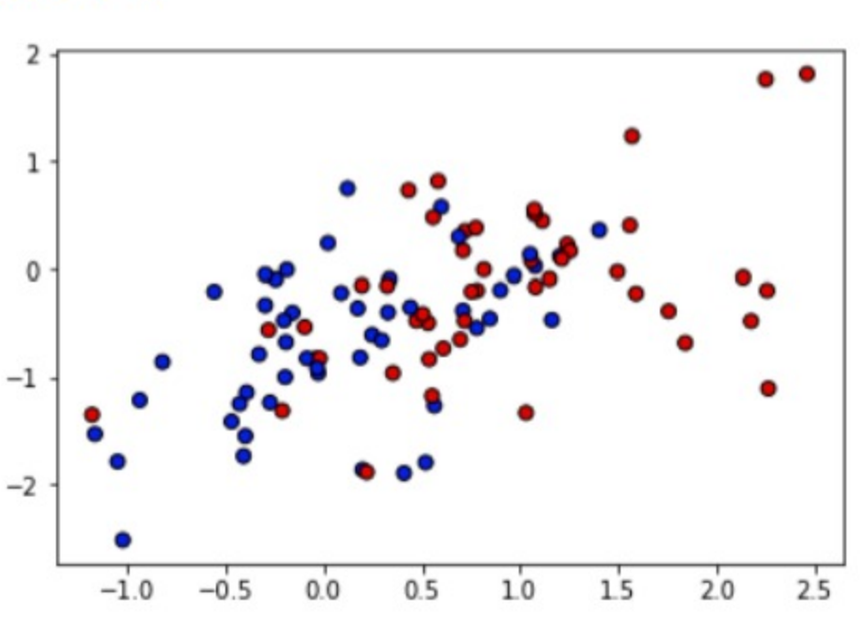
In [96]:
```python
print()
print("\033[1m" + "Dataset A:")
ml.plotClassify2D(None, XA, YA, edgecolor = "black")
```

Dataset A:



In [97]:
```python
print()
print("\033[1m" + "Dataset B:")
ml.plotClassify2D(None, XB, YB, edgecolor = "black")
```

Dataset B:

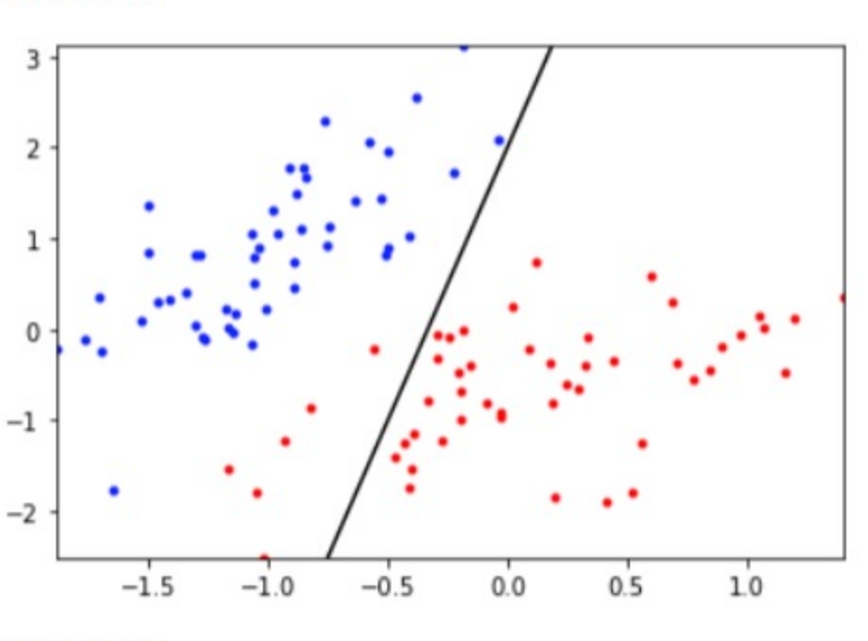


Dataset A is linearly separable.

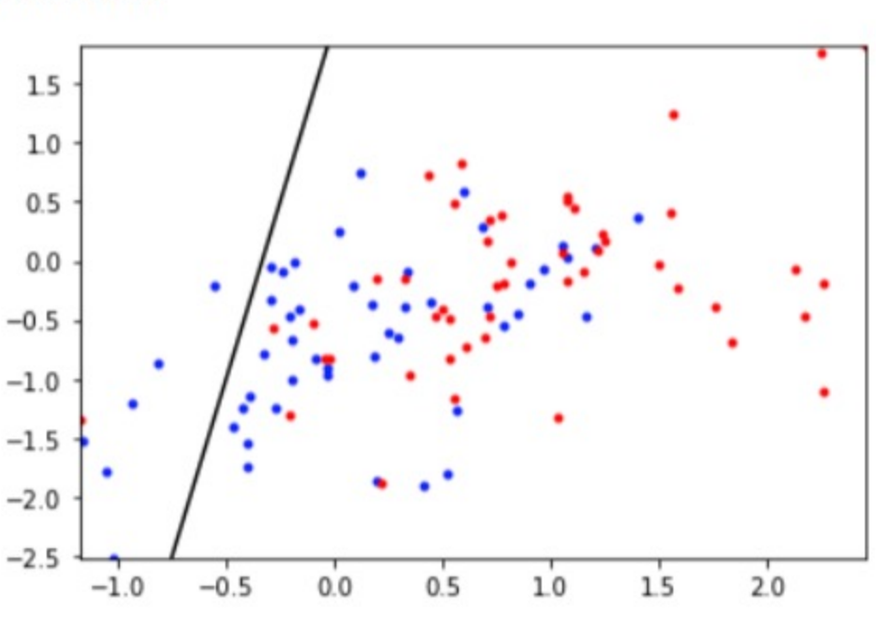
Problem 1 - Part 2:

In [98]:
```python
def myPlotBoundary(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]]);
    ax0 = -(self.theta[0] + self.theta[1] * x1b[0])/ self.theta[2]
    ax1 = -(self.theta[0] + self.theta[1] * x1b[1])/ self.theta[2]
    x2b = np.array([ax0,ax1]);
    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();

class logisticClassify2(ml.classifier):
    classes = []
    theta = np.array( [-1, 0, 0] )
    plotBoundary = myPlotBoundary
    predict = None
    train = None

print()
print("\033[1m" + "Dataset A:")
learnerA = logisticClassify2()
learnerA.classes = np.unique(YA)
learnerA.theta = np.array([2, 6, -1])
learnerA.plotBoundary(XA,YA)
plt.show()

print()
print("\033[1m" + "Dataset B:")
learnerB = logisticClassify2()
learnerB.classes = np.unique(YB)
learnerB.theta = np.array([2, 6, -1])
learnerB.plotBoundary(XB,YB)
plt.show()
```

Dataset A:



Dataset B:



Problem 1 - Part 3:

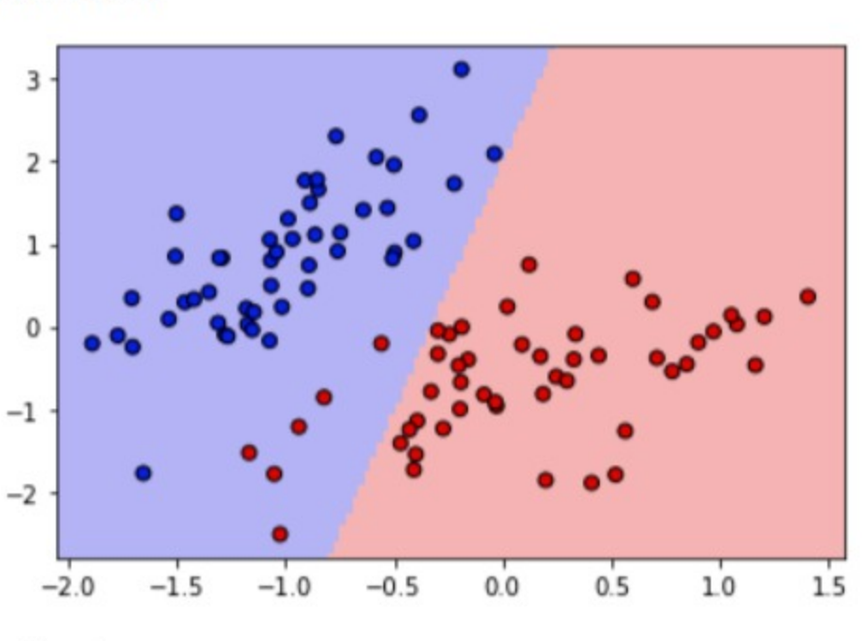
In [99]:
```python
def myPredict(self,X):
    temp = [0] * X.shape[0]
    Yhat = [0] * X.shape[0]
    for k in range(0, X.shape[0]):
        temp[k] = self.theta[0] + self.theta[1] * X[k, 0] + self.theta[2] * X[k, 1]
        if(temp[k] > 0):
            Yhat[k] = self.classes[1]
        else:
            Yhat[k] = self.classes[0]
    Yhat = np.array(Yhat)
    return Yhat

class logisticClassify2(ml.classifier):
    classes = []
    theta = np.array( [-1, 0, 0] )
    plotBoundary = myPlotBoundary
    predict = myPredict
    train = None

print()
learnerA = logisticClassify2()
learnerA.classes = np.unique(YA)
learnerA.theta = np.array([2, 6, -1])
print("\033[1m" + "Error for Dataset A: ", learnerA.err(XA,YA), "\n")

learnerB = logisticClassify2()
learnerB.classes = np.unique(YB)
learnerB.theta = np.array([2, 6, -1])
print("\033[1m" + "Error for Dataset B: ", learnerB.err(XB,YB))
```

```
Error for Dataset A:  0.06060606060606061

Error for Dataset B:  0.45454545454545453
```

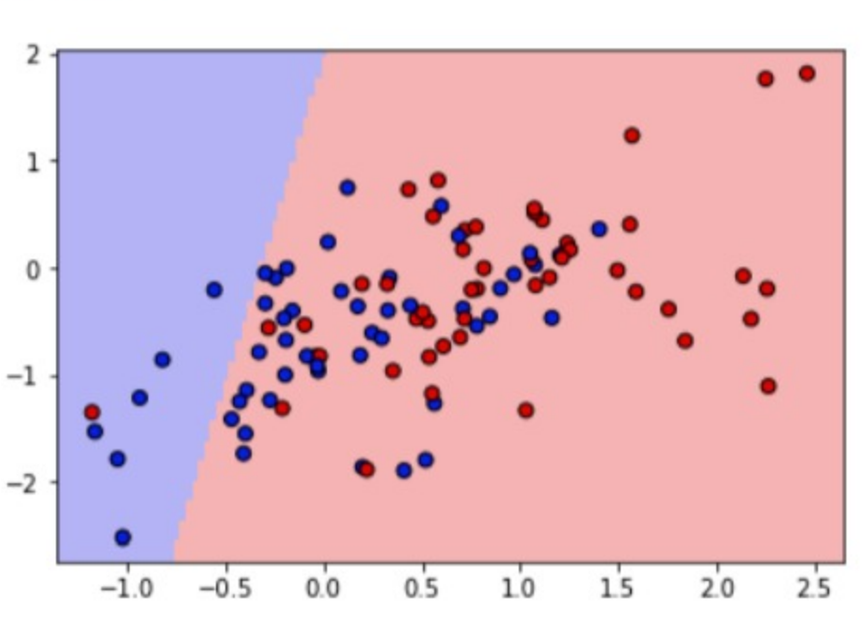
Problem 1 - Part 4:

In [100]:
```python
print()
print("\033[1m" + "Dataset A:")
learnerA.classes = np.unique(YA)
ml.plotClassify2D(learnerA, XA, YA, edgecolor = "black")
plt.show()
print()
print("\033[1m" + "Dataset B:")
learnerB.classes = np.unique(YB)
ml.plotClassify2D(learnerB, XB, YB, edgecolor = "black")
plt.show()
```

Dataset A:



Dataset B:



Problem 1 - Part 5:

In [101]:
```python
import IPython.display
print()
IPython.display.Image("temp.jpg")
```

Out[101]:

For a single data point j the logistic negative - log-likehood loss is :

$$J_j(\theta) = -y^{(j)} \log \sigma(x^{(j)}.\theta) - (1-y^{(j)}) \log (1-\sigma(x^{(j)}.\theta))$$

$$\theta = \{\theta_0, \theta_1, \theta_2\}$$

$$\frac{\partial (x^{(j)}.\theta)}{\partial \theta_2} = \frac{\partial (\theta_0 + \theta_1 x_1^{(j)} + \theta_2 x_2^{(j)})}{\partial \theta_2} = x_2^{(j)}$$

$$\frac{d\sigma(z)}{dz} = \sigma(z)(1-\sigma(z)) \Rightarrow$$

$$\sigma'(x^{(j)}.\theta) = \sigma(x^{(j)}.\theta)(1-\sigma(x^{(j)}.\theta))$$

$$\frac{\partial J_j(\theta)}{\partial \theta_2} = (-1) y^{(j)} \left[\frac{1}{\sigma(x^{(j)}.\theta)} \times \sigma'(x^{(j)}.\theta)\, x_2^{(j)}\right] - (-y^{(j)}-1)\left[\frac{1}{1-\sigma(x^{(j)}.\theta)} \times \sigma'(x^{(j)}.\theta)\, x_2^{(j)}\right] \Rightarrow$$

$$\Rightarrow \begin{cases} \frac{\partial J_j(\theta)}{\partial \theta_0} = (-1) y^{(j)} (1-\sigma(x^{(j)}.\theta)) + (1-y^{(j)})\sigma(x^{(j)}.\theta) \\ \frac{\partial J_j(\theta)}{\partial \theta_1} = (-1) y^{(j)} [(1-\sigma(x^{(j)}.\theta)) x_1^{(j)}] - (1-y^{(j)})(\sigma(x^{(j)}.\theta) x_1^{(j)}) \\ \frac{\partial J_j(\theta)}{\partial \theta_2} = (-1) y^{(j)} [(1-\sigma(x^{(j)}.\theta)) x_2^{(j)}] - (1-y^{(j)})(\sigma(x^{(j)}.\theta) x_2^{(j)}) \end{cases}$$

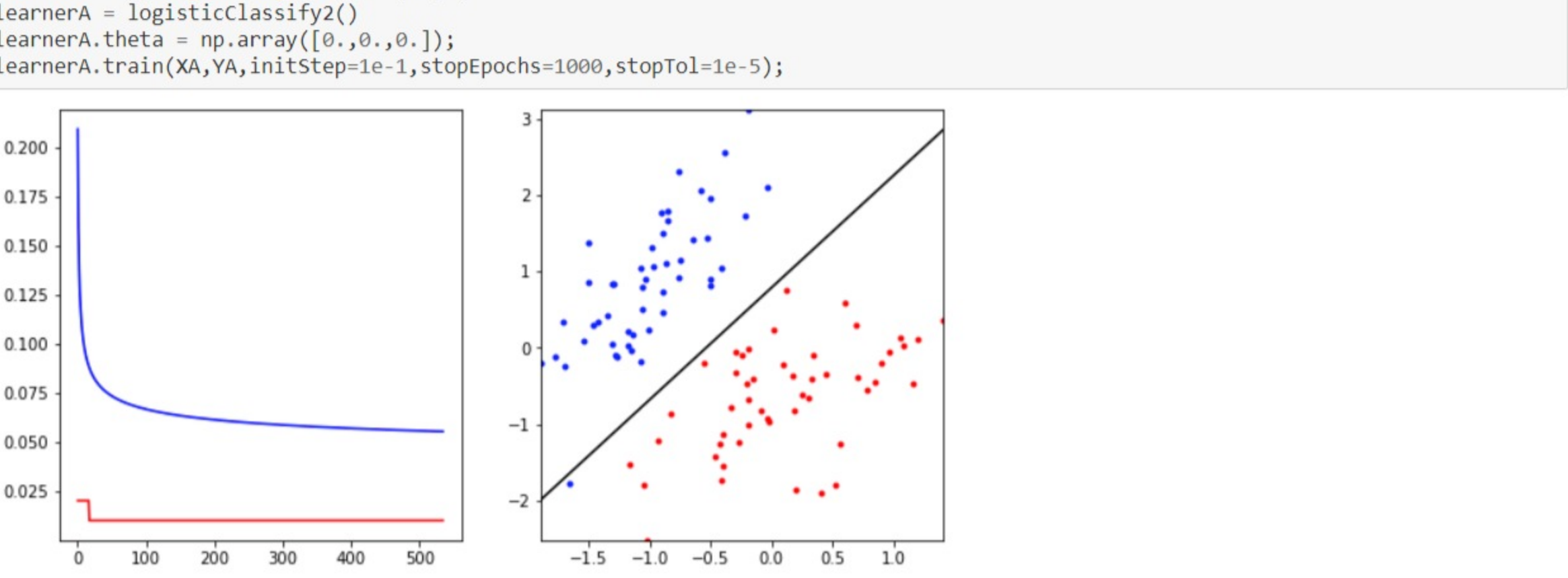
Problem 1 - Part 6:

In [102]:
```python
def myTrain(self, X, Y, initStep=1.0, stopTol=1e-4, stopEpochs=5000, plot=None):
    from IPython import display
    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    = 1.0 / (1.0 + np.exp(-(XX[i, :].dot(self.theta))))
            gradi = -YY[i] * (1 - ri) * XX[i, :] + (1 - YY[i]) * XX[i,:] * ri
            self.theta -= stepsize * gradi;
        J01.append( self.err(X,Y) )
        temp = 1.0 / (1.0 + np.exp(-(XX.dot(self.theta))))
        Jnll.append((-1) * np.mean(YY * np.log(temp) + (1 - YY) * np.log(1 - temp)))
        display.clear_output(wait=True);
        plt.subplot(1,2,1); plt.cla(); plt.plot(Jnll,'b-',J01,'r-');
        if N==2: plt.subplot(1,2,2); plt.cla(); self.plotBoundary(X,Y);
        plt.pause(.01);
        done = ((epoch > 1) and (np.abs(Jnll[-1] - Jnll[-2]) < stopTol) or (stopEpochs < epoch))
```

Problem 1 - Part 7:

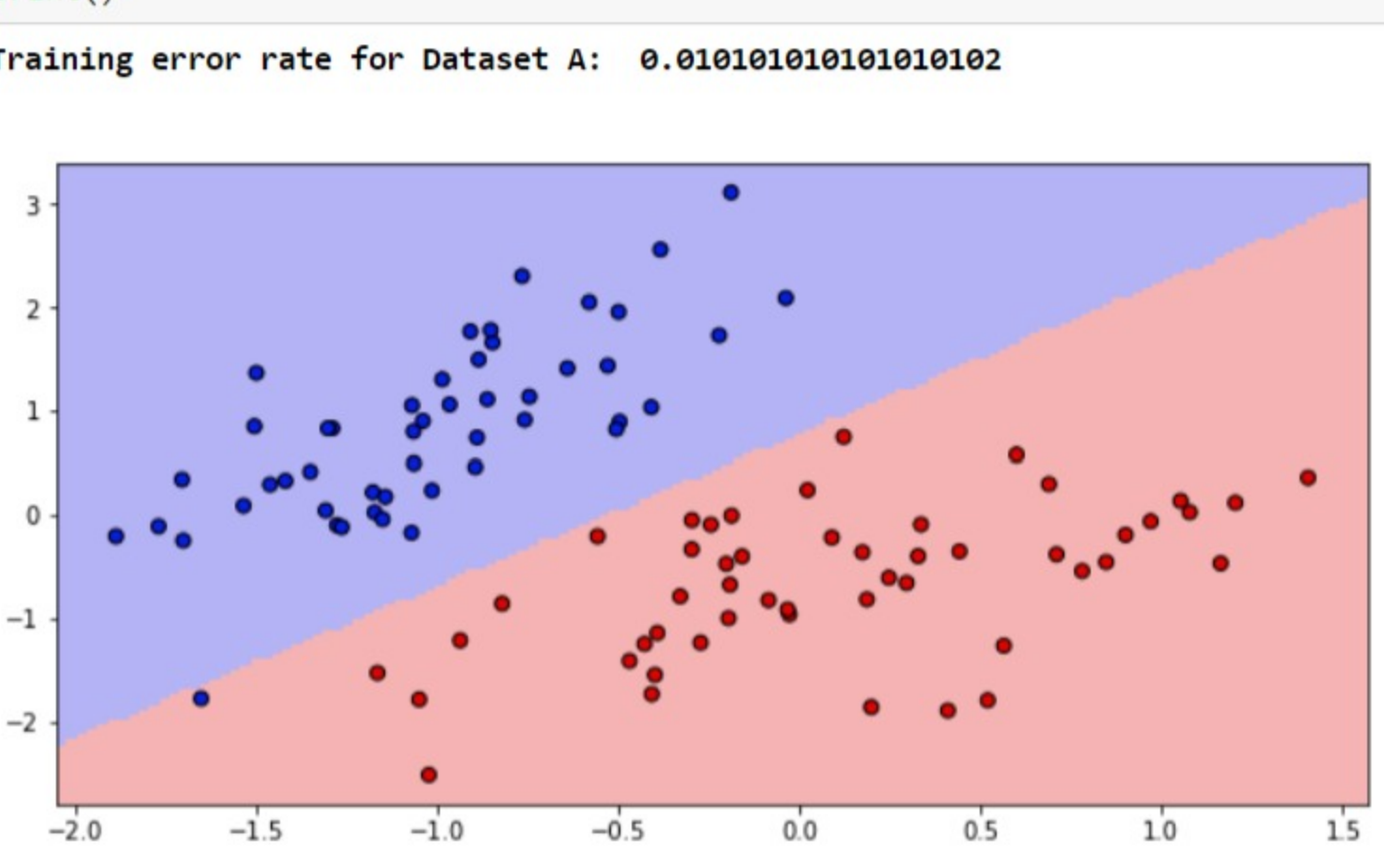
Dataset A:

In [103]:
```python
class logisticClassify2(ml.classifier):
    classes = []
    theta = np.array( [-1, 0, 0] )
    plotBoundary = myPlotBoundary
    predict = myPredict
    train = myTrain

plt.rcParams['figure.figsize'] = (10,5)
learnerA = logisticClassify2()
learnerA.theta = np.array([0.,0.,0.]);
learnerA.train(XA,YA,initStep=1e-1,stopEpochs=1000,stopTol=1e-5);
```



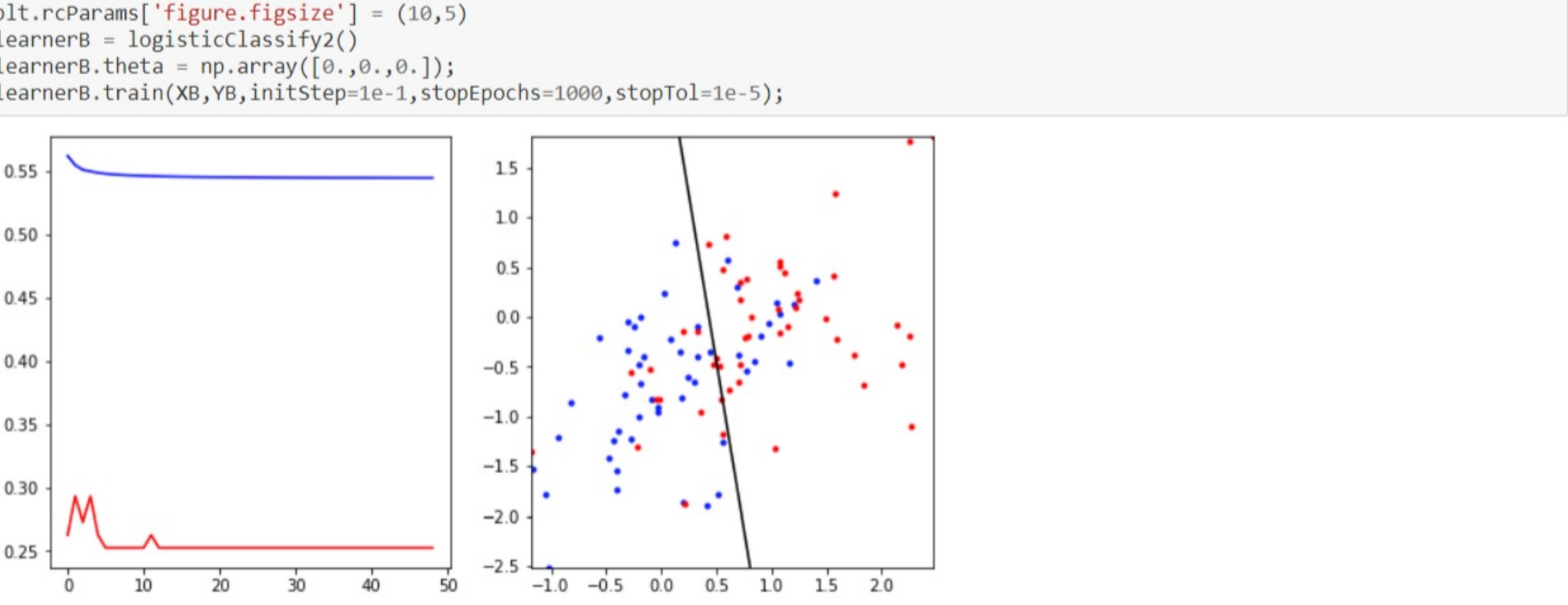
In [104]:
```python
ml.plotClassify2D(learnerA,XA,YA,edgecolor = "black")
print()
print("\033[1m" + "Training error rate for Dataset A: ",learnerA.err(XA,YA), "\n")
plt.show()
print()
```

```
Training error rate for Dataset A:  0.010101010101010102
```



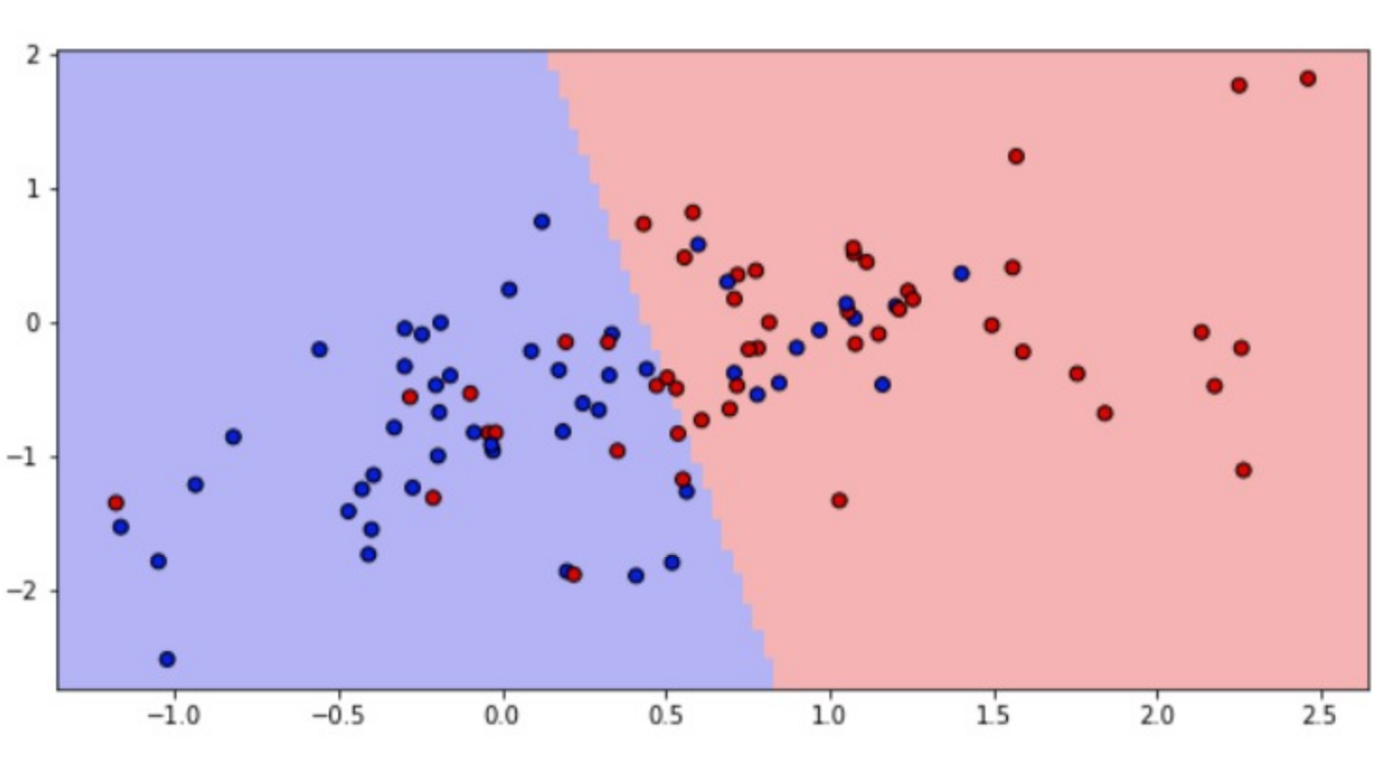
Dataset B: ¶

In [120]:
```python
class logisticClassify2(ml.classifier):
    classes = []
    theta = np.array( [-1, 0, 0] )
    plotBoundary = myPlotBoundary
    predict = myPredict
    train = myTrain

plt.rcParams['figure.figsize'] = (10,5)
learnerB = logisticClassify2()
learnerB.theta = np.array([0.,0.,0.]);
learnerB.train(XB,YB,initStep=1e-1,stopEpochs=1000,stopTol=1e-5);
```



In [115]:
```python
ml.plotClassify2D(learnerB,XB,YB,edgecolor = "black")
print()
print("\033[1m" + "Training error rate for Dataset B: ",learnerB.err(XB,YB), "\n")
plt.show()
print()
```

```
Training error rate for Dataset B:  0.25252525252525254
```



Problem 2 - Part 1:

Datasets which can be shatterred by learner 1 are: a, b

The datasets a and b can be shattered by this learner since the boundry decision line will be a vertical line.

VC dimension of the classifier: 2

Problem 2 - Part 2:

Datasets which can be shattered by learner 2 are: a, b, c

For d, we cannot find any line that separates the two classes and goes throught the (0, 0) which is the origin in this case by choosing points (2, 2) (8, 6) to be the same class.

VC dimension of the classifier: 3

Problem 2 - Part 3:

Datasets which can be shattered by learner 3 are: a, b, c

For d, by choosing points (6, 4) (4, 8) in order to be the same class we cannot find any classification that separates the two classes since any circle that counts two points as class one, cannot keeps other two points in the circle for a different class two.

VC dimension of the classifier: 3

Problem 2 - Part 4:

Datasets which can be shattered by learner 4 are: a, b, c, d

In this learner we have two parallel lines which can shatter all four datasets.

VC dimension of the classifier: 4

Problem 3:

I completed HW3 entirely on my own by using the lecture videos and the discussion sessions. In addition, I used the the HW3_Template which was provided as the starting point for this homework and read through the base codes in .py file which was provided in the zip file. Finally, I completely followed the academic honesty guidelines which is on our canvas website and I did not discuss my homework with anyone in-person.

Problem 4:

For Halloween, I was home all the time and I was studying for my midterms since I have 20 units this quarter which is my last quarter in UCI.