Example Template for HW3

This notebook contains the same template code as "logisticClassify2.py", but reorganized to make it simpler to edit and solve in iPython. Feel free to use this for your homework, or do it another way, as you prefer.

In [1]:
```
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   # use matplotlib for plotting with in
plt.set_cmap('jet');
%matplotlib inline
import warnings
warnings.filterwarnings('ignore'); # for deprecated matplotlib function

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)       # reorder randomly rather than by clas

X,_  = ml.transforms.rescale(X)  # rescale to improve numerical stabili

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


```

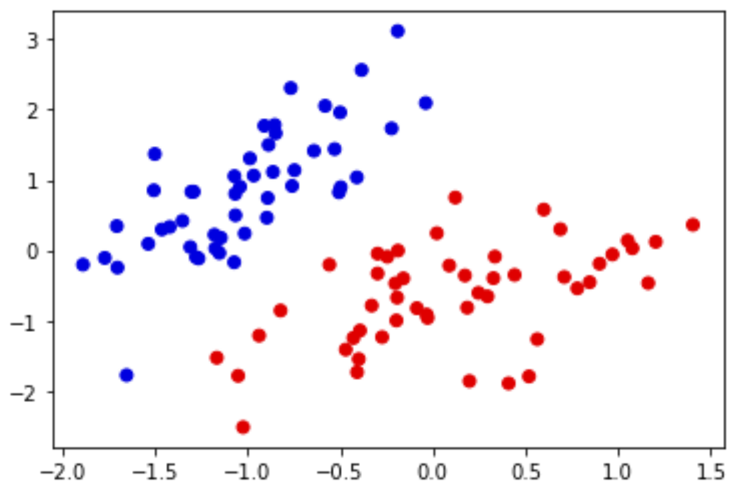
P1.1

For each of the two datasets, create a separate scatter plot in which the training data from the two classes is plotted in different colors. Which of the two datasets is linearly separable?

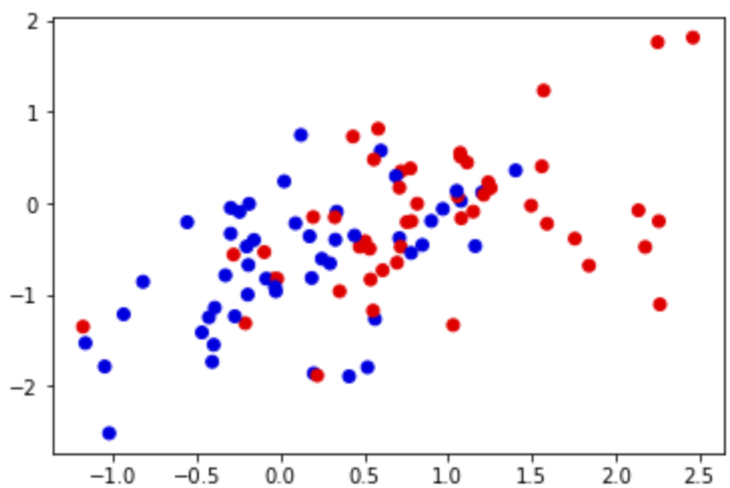
P1.1 Answer : dataset set A can linearly

separable. But some of data in set B is overlapping so too hard to be linearly separable

In [3]:
```
ml.plotClassify2D( None, XA, YA)
plt.show()
```



In [4]:
```
ml.plotClassify2D( None, XB, YB)
plt.show()
```



P1.2

Write (fill in) the function plotBoundary in logisticClassify2.py to compute the points on the decision boundary. In particular, you only need to make sure x2b is set correctly using self.theta . This will plot the data & boundary quickly, which is useful for visualizing the model during training. To demonstrate your function, plot the decision boundary corresponding to the classifier

```
sign( 2 + 6*X1 - 1*X2 )
```

along with dataset A, and again with dataset B. These fixed parameters should lead to an OK classifier on one data set, but a poor classifier on the other. You can create a “blank” learner and set the weights as follows:

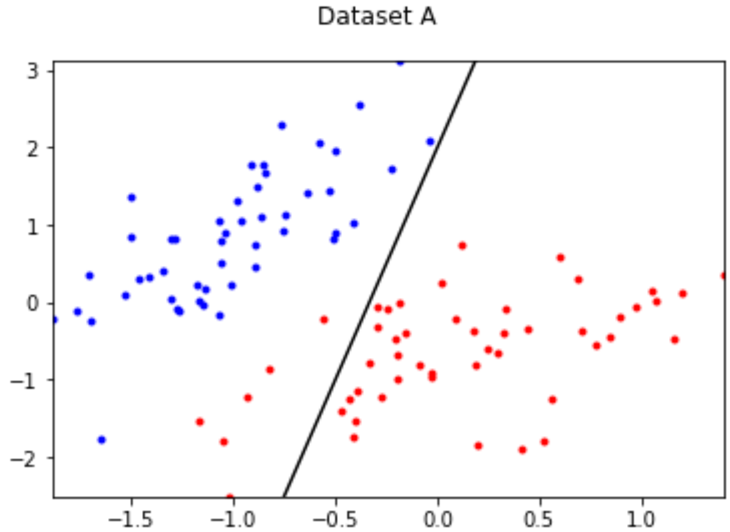
Set the weights as follows:

In [5]:

```
def myPlotBoundary(self, X,Y):




    """ Plot the (linear) decision boundary of the classifier, along wi
    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]);

    ## TODO: find points on decision boundary defined by theta0 + theta
    x1b = np.array([ax[0],ax[1]]);    # at X1 = points in x1b # HW3 giv

    # TODO find x2 values as a function of x1's values
    # x2b = np.array([-(self.theta[0]+self.theta[1]*x1b[0])/self.theta[
    x2b = (-self.theta[0]-self.theta[1]*x1b)/self.theta[2]

    ## Now plot the data and the resulting boundary:
    A = Y==self.classes[0]; # and plot it: # HW3 given code
    plt.plot(X[A,0],X[A,1],'b.',X[~A,0],X[~A,1],'r.',x1b,x2b,'k-'); # H
    plt.axis(ax);
    plt.draw();

# Create a shell classifier
class logisticClassify2(ml.classifier):
    classes = []
    theta = np.array( [-1, 0, 0] )   # initialize theta to something #
    plotBoundary = myPlotBoundary
    predict = None                   # these functions will be implemen
    train = None
```

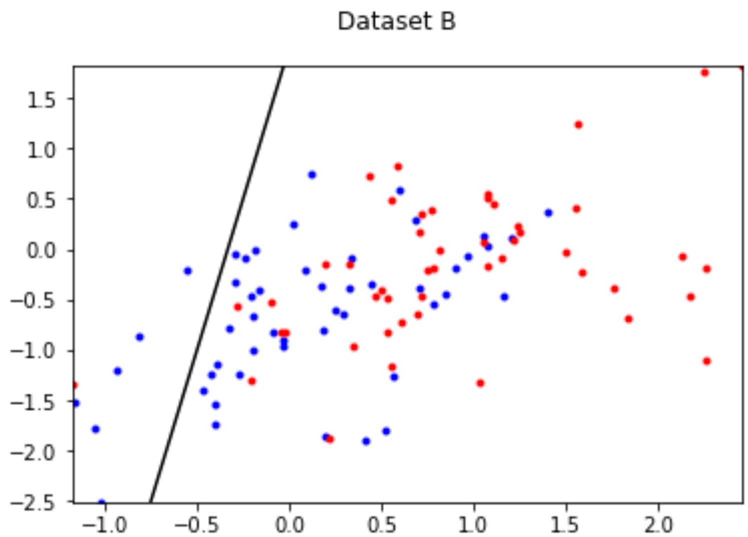
In [6]:

```
learnerA = logisticClassify2()
learnerA.classes = np.unique(YA)        # store the class values for thi
learnerA.theta = np.array([2,6,-1]);    # TODO: insert hard-coded values
learnerA.plotBoundary(XA,YA)

plt.suptitle('Dataset A')
plt.figure(figsize=(10,5))
plt.show()
```



<Figure size 720x360 with 0 Axes>

In [7]:
```
learnerA = logisticClassify2()# HW3 given code
learnerA.classes = np.unique(YB)        # store the class values for thi
learnerA.theta = np.array([2,6,-1]);    # TODO: insert hard-coded values
learnerA.plotBoundary(XB,YB)

plt.suptitle('Dataset B')
plt.figure(figsize=(10,5))
plt.show()
```




```
<Figure size 720x360 with 0 Axes>
```

In [36]:
```
# ...

```

P1.3 : predict function and error rate

In [34]:

```
# Should go in your logistic2 class:
def myPredict(self,X):
    """ Return the predictied class of each data point in X"""
    """
    raise NotImplementedError

    ## TODO: compute linear response r[i] = theta0 + theta1 X[i,1] + th

    ##       else predict class 0:  Yhat[i] = self.classes[0]
    """
    leng = len(X)

    errRate = np.zeros(leng)
    Yhat = np.zeros(leng)

    ## TODO: compute linear response r[i] = theta0 + theta1 X[i,first]
    for i in range ( leng ):
        errRate[i] = self.theta[0] + self.theta[1]*X[i,0 ] + self.theta

        ## TODO: if r[i] > 0, predict class 1:  Yhat[i] = self.classes[
        if (errRate[i] > 0):
            Yhat[i] = self.classes[1]
        ## TODO: else predict class 0:  Yhat[i] = self.classes[0]
        else:
            Yhat[i] = self.classes[0]
    return Yhat


# Update our shell classifier definition
class logisticClassify2(ml.classifier):
    classes = [] # HW3 given code
    theta = np.array( [-1, 0, 0] )   # initialize theta to something# H
    plotBoundary = myPlotBoundary
    predict = myPredict
    train = None# HW3 given code



# ...
```

In [35]:

```
learnerA = logisticClassify2()
learnerA.classes = np.unique(YA)        # store the class values for th
learnerA.theta = [2,6,-1];   # TODO: insert hard-coded values# HW3 give

print ( "For Dataset A ")
print( "Error: ", learnerA.err(XA,YA) )
```

```
For Dataset A
Error:  0.06060606060606061
```

In [36]:
```
learnerA = logisticClassify2()
learnerA.classes = np.unique(YB)        # store the class values for th
learnerA.theta = [2,6,-1];    # TODO: insert hard-coded values# HW3 give

print ( "For Dataset B ")
print( "Error: ", learnerA.err(XB,YB) )
```

```
For Dataset B
Error:  0.45454545454545453
```

If predict is implemented, then the inherited 2D visualization function should work; you can verify your decision boundary from P1.2:

1.4

Verify that your predict and plotBoundary implementations are consistent by using plotClassify2D with your manually constructed learner on each dataset. This will call predict on a dense grid of points, and you should find that the resulting decision boundary matches the one you plotted previously. (5 points)

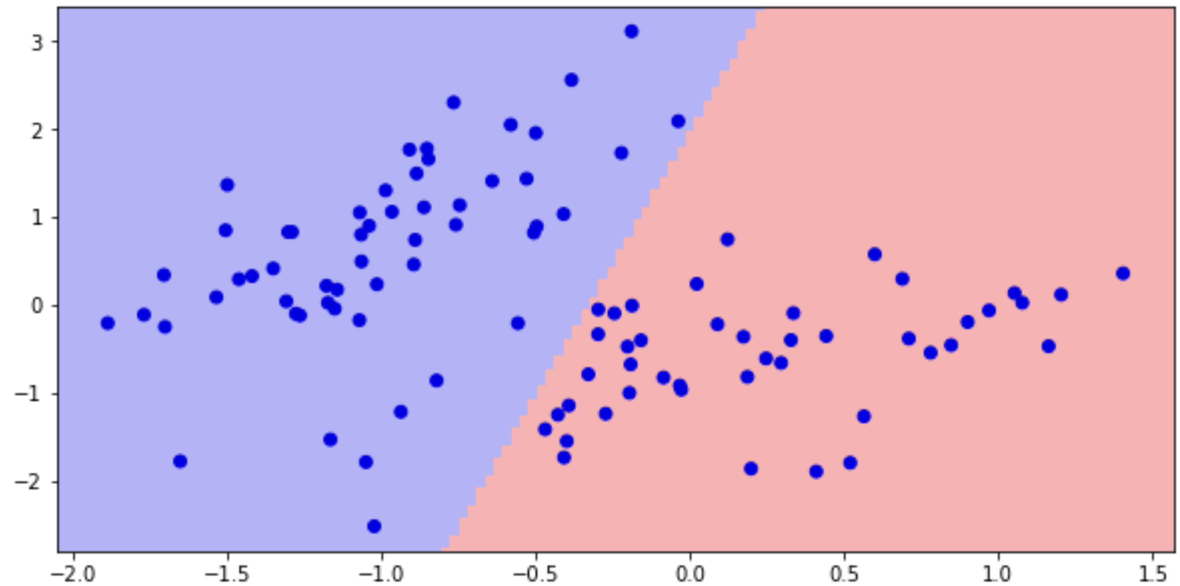
- plotBoundary

1.4 answer

- Although some of the points in Dataset A have crossed the decision boundary, generally given theta are well applied to Dataset A.
- In Dataset B, the decision boundary does not function properly. Given thetaes do not apply well to B

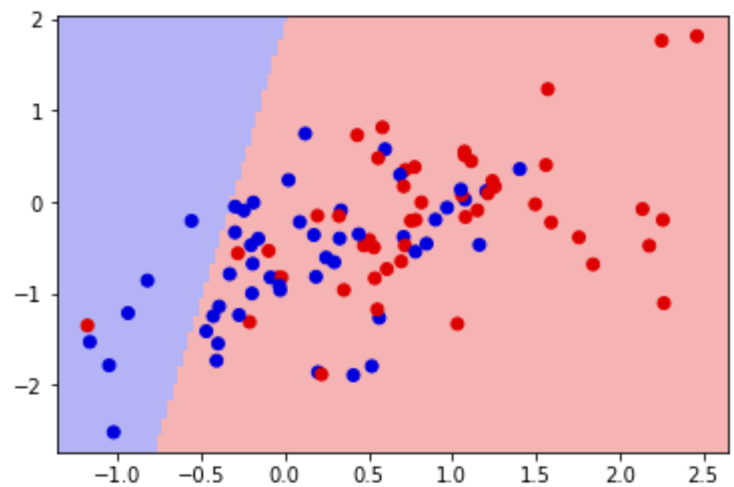
In [37]:

```
ml.plotClassify2D(learnerA,XA,YA)
plt.show()
```



In [12]:

```
ml.plotClassify2D(learnerA,XB,YB)
plt.show()
```



...

Here is an example of latex equations that may be useful for expressing the gradient:

1.5 Gradient of NLL

Our negative log-likelihood loss is:

$$J_j(\theta) = -\begin{cases} \log(\sigma(x^{(i)} \cdot \theta)) & \text{if } y^{(i)} = 1 \\ \log(1 - \sigma(x^{(i)} \cdot \theta)) & \text{if } y^{(i)} = 0 \end{cases}$$

Thus, its gradient is:

$$\nabla J_j(\theta) = (something)$$

- $J_j(\theta) = -y^{(j)} \log \alpha \cdot (x^j \cdot \theta) - (1-y^j) \cdot \log\left(1 - \alpha(x^j \cdot \theta)\right)$

- $\nabla J_j = (\alpha(x^i \cdot \theta) - y^j)\ x^{(j)}$

- $\alpha'(r) = \alpha(r) \cdot (1 - \alpha(r))$

$$
\begin{aligned}
\nabla J_i(\theta) &= -y^{(i)} \cdot (1 - s^i) \cdot x^i + (1 - y^i) \cdot s^i \cdot x^i \\
&= -y^i \cdot x^i + y^i \cdot s^i \cdot x^i + s^i \cdot x^i - y^i \cdot s^i \cdot x^i \\
&= -y^i \cdot x^i + s^i \cdot x^i = (-y^i + s^i) x^i
\end{aligned}
$$

1.6

Now define the train function and complete its missing code.

In [52]:

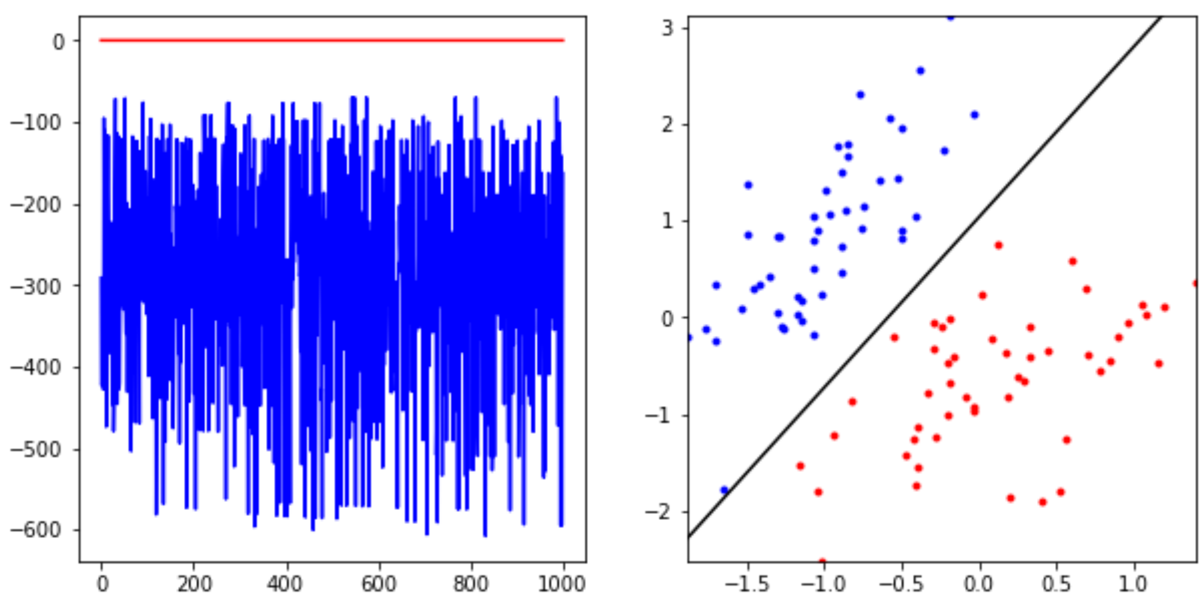
```
def myTrain(self, X, Y, initStep=1.0, stopTol=1e-4, stopEpochs=5000, pl
    """ Train the logistic regression using stochastic gradient descent
    from IPython import display
    M,N = X.shape;                     # initialize the model if necess
    self.classes = np.unique(Y);       # Y may have two classes, any va
    XX = np.hstack((np.ones((M,1)),X)) # XX is X, but with an extra col
    YY = ml.toIndex(Y,self.classes);   # YY is Y, but with canonical va
    if len(self.theta)!=N+1: self.theta=np.random.rand(N+1);
    # init loop variables:
    epoch=0; done=False; Jnll=[]; J01=[];
    while not done:
        stepsize, epoch = initStep*2.0/(2.0+epoch), epoch+1; # update s
        # Do an SGD pass through the entire data set:

        """
        # TODO: compute linear response r(x)
        # TODO: compute gradient of NLL loss
        # take a gradient step
        """
        for i in np.random.permutation(M):
            ri = XX[i].dot(self.theta)
            si    = 1/ ( 1+np.exp(-ri));     # TODO: compute linear resp

            gradi = (si - YY[i])*XX[i];      # TODO: compute gradient of
            self.theta -= stepsize * gradi;  # take a gradient step

        J01.append( self.err(X,Y) )  # evaluate the current error rate

        """
        ## TODO: compute surrogate loss (logistic negative log-likeliho
        ##  Jsur = - sum_i [ (log si) if yi==1 else (log(1-si)) ]
        """
        Jsur = 0;
        for i in range (M):

            if ( YY[i] == 1):
                Jsur = Jsur + np.log(si)
            else:
                Jsur = Jsur + np.log(1-si)

        J = np.mean(Jsur)


        Jnll.append( J ) # TODO evaluate the current NLL loss
        display.clear_output(wait=True);
        plt.subplot(1,2,1);
        plt.cla();
        plt.plot(Jnll,'b-',J01,'r-'); # plot losses
        if N==2: plt.subplot(1,2,2); plt.cla(); self.plotBoundary(X,Y);
        plt.pause(.01);                       # let OS draw the plot

        """
        ## For debugging: you may want to print current parameters & lo
        # print self.theta, ' => ', Jnll[-1], ' / ', J01[-1]
        # raw_input()   # pause for keystroke

```

```
        # TODO check stopping criteria: exit if exceeded # of epochs (
        """
#         done = NotImplementedError;   # or if Jnll not changing betwee
        done = epoch>=stopEpochs or (epoch>1 and abs(Jnll[-1]-Jnll[-2])
```

1.7

In [53]:

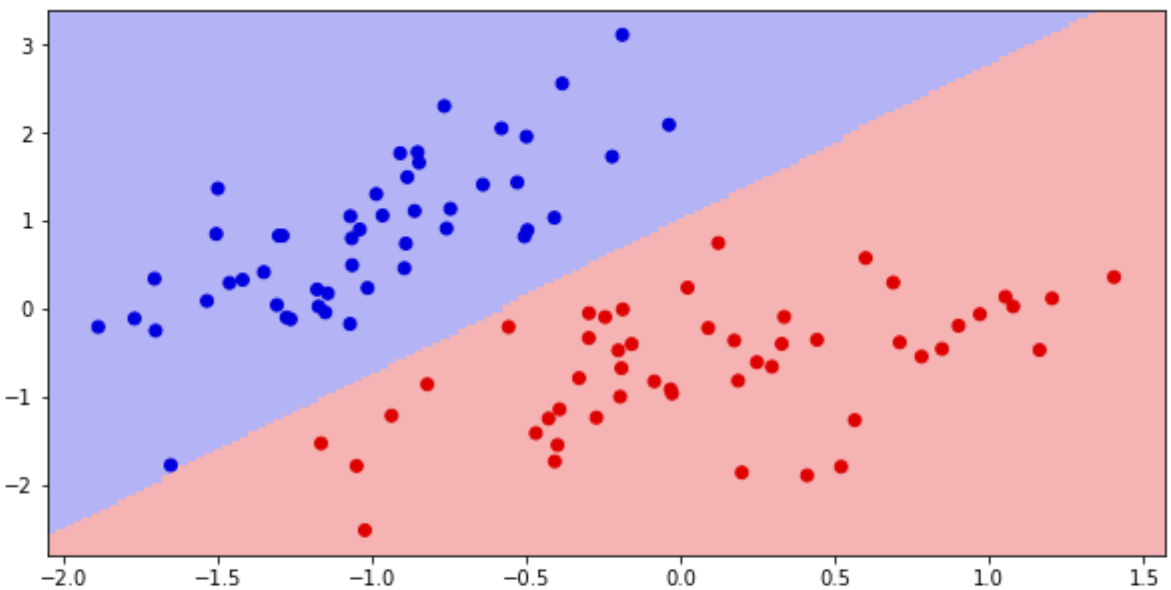
```
# Update our shell classifier definition
class logisticClassify2(ml.classifier):
    classes = []
    theta = np.array( [-1, 0, 0] )   # initialize theta to something
    plotBoundary = myPlotBoundary    #
    predict = myPredict              # Now all parts are implemented
    train = myTrain

plt.rcParams['figure.figsize'] = (10,5)    # make a wide figure, for tw

learnerA = logisticClassify2()
#learnerA.theta = np.array([0.,0.,0.]);
learnerA.theta = np.array([2, 6, -1.]);
learnerA.train(XA,YA,initStep=1e-1,stopEpochs=1000,stopTol=1e-5);

```



In [18]:

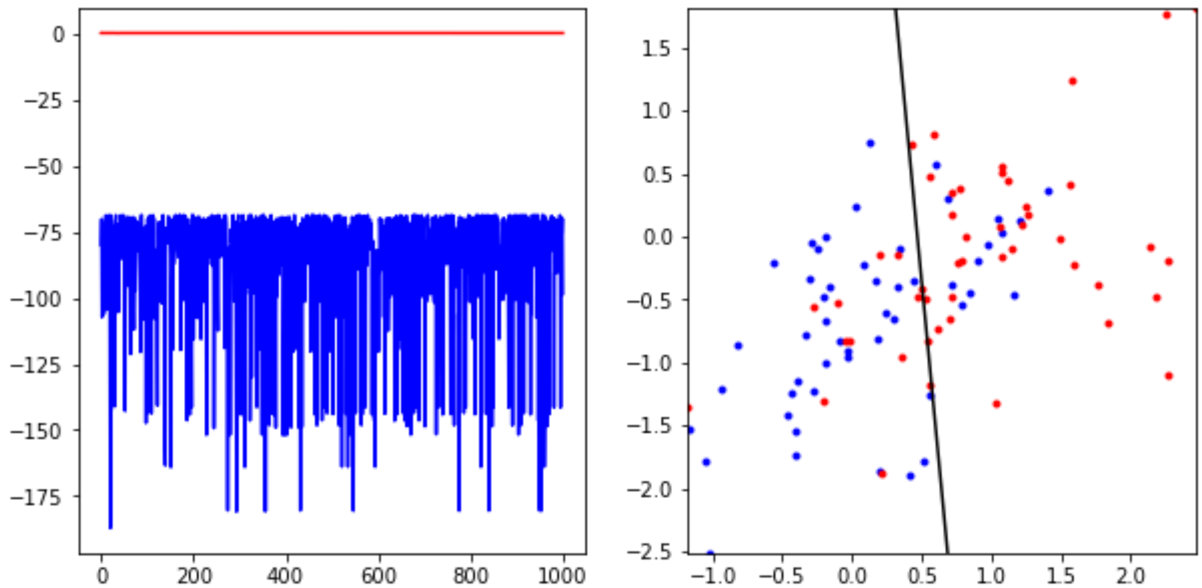
```
ml.plotClassify2D(learnerA,XA,YA)
print("Training error rate: ",learnerA.err(XA,YA))

plt.show()
```

```
Training error rate:  0.0
```



In [59]:

```

plt.rcParams['figure.figsize'] = (10,5)
learnerB = logisticClassify2()
learnerB.theta = np.array([2, 6, -1.]);
learnerB.train(XB,YB,initStep=1.,stopEpochs=1000,stopTol=1e-5);
```

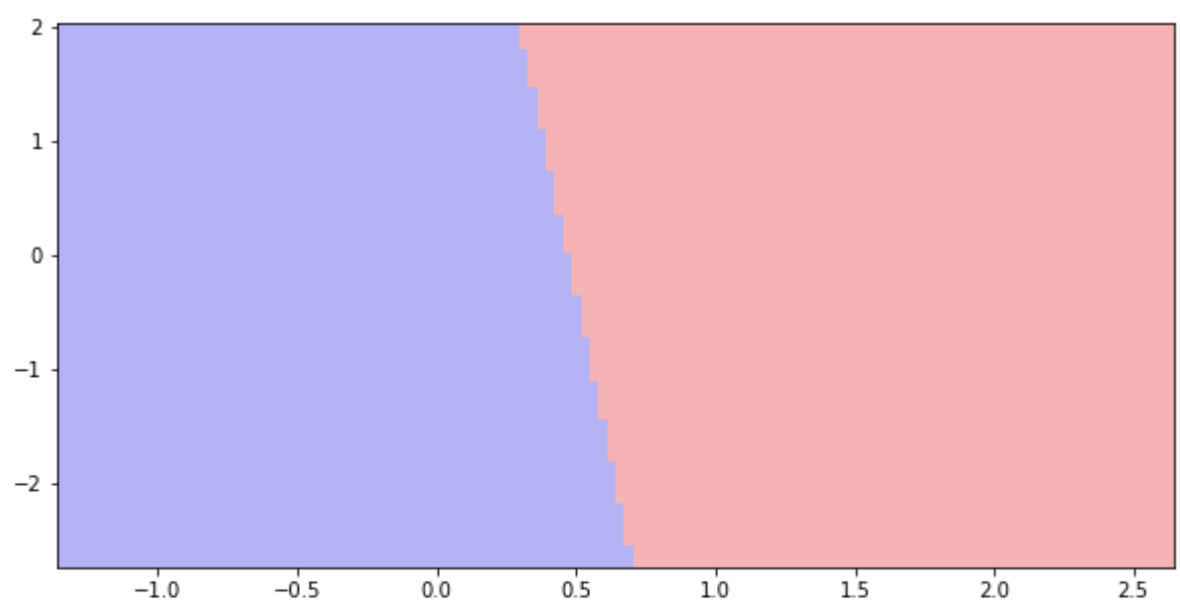


In [65]:

```python
ml.plotClassify2D(learnerB,XB,XB)
print("Training error rate: ",learnerB.err(XB,XB))

plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~/opt/anaconda3/lib/python3.8/site-packages/matplotlib/axes/_axes.py in _parse_scatter_color_args(c, edgecolors, kwargs, xsize, get_next_color_func)
   4238             try:  # Is 'c' acceptable as PathCollection facecolors?
-> 4239                 colors = mcolors.to_rgba_array(c)
   4240             except ValueError:

~/opt/anaconda3/lib/python3.8/site-packages/matplotlib/colors.py in to_rgba_array(c, alpha)
    306         if np.any((result < 0) | (result > 1)):
--> 307             raise ValueError("RGBA values should be within 0-1 range")
    308         return result

ValueError: RGBA values should be within 0-1 range

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
<ipython-input-65-cc055c138595> in <module>
----> 1 ml.plotClassify2D(learnerB,XB,XB)
      2 print("Training error rate: ",learnerB.err(XB,XB))
      3 
      4 plt.show()

~/CS178/178-hw3-code/mltools/plot.py in plotClassify2D(learner, X, Y, pre, ax, nGrid, cm, bgalpha, soft, **kwargs)
     72     if len(Y.shape)==1 or Y.shape[1]==1: data_colors = classcolor[np.searchsorted(classes,Y)];  # use colors if Y is discrete class
     73     else: data_colors = Y.dot(classcolor); data_colors[data_colors>1]=1;          # blend colors if Y is a soft confidence
---> 74     ax.scatter(X[:,0],X[:,1], c=data_colors, **kwargs);
     75 
     76     #for i,c in enumerate(classes):                  # old code: used plot instead of scatter

~/opt/anaconda3/lib/python3.8/site-packages/matplotlib/__init__.py in inner(ax, data, *args, **kwargs)
   1563     def inner(ax, *args, data=None, **kwargs):
   1564         if data is None:
-> 1565             return func(ax, *map(sanitize_sequence, args), **kwargs)
   1566 
   1567         bound = new_sig.bind(ax, *args, **kwargs)

~/opt/anaconda3/lib/python3.8/site-packages/matplotlib/cbook/deprecation.
```

```
py in wrapper(*args, **kwargs)
    356                     f"%(removal)s.  If any parameter follows {name!
r}, they "
    357                     f"should be pass as keyword, not positionally.")
--> 358             return func(*args, **kwargs)
    359 
    360         return wrapper

~/opt/anaconda3/lib/python3.8/site-packages/matplotlib/axes/_axes.py in s
catter(self, x, y, s, c, marker, cmap, norm, vmin, vmax, alpha, linewidth
s, verts, edgecolors, plotnonfinite, **kwargs)
   4399 
   4400         c, colors, edgecolors = \
-> 4401             self._parse_scatter_color_args(
   4402                 c, edgecolors, kwargs, x.size,
   4403                 get_next_color_func=self._get_patches_for_fill.ge
t_next_color)

~/opt/anaconda3/lib/python3.8/site-packages/matplotlib/axes/_axes.py in _
parse_scatter_color_args(c, edgecolors, kwargs, xsize, get_next_color_fun
c)
   4240             except ValueError:
   4241                 if not valid_shape:
-> 4242                     raise invalid_shape_exception(c.size, xsize)
   4243                 # Both the mapping *and* the RGBA conversion fail
ed: pretty
   4244                 # severe failure => one may appreciate a verbose
 feedback.

ValueError: 'c' argument has 396 elements, which is inconsistent with 'x'
and 'y' with size 99.
```



In [ ]:

```
1
```

Problem 3: Statement of Collaboration (5 points)

```
Gwenth : discus , Shattering and VC Dimension
Piazza : 359
Park : discus regression
```

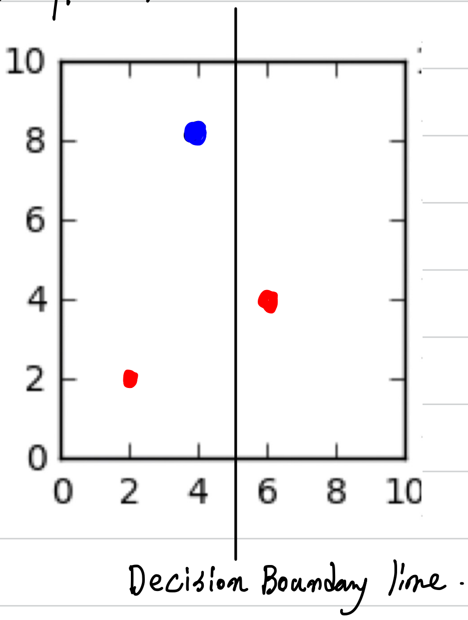
▪▪▪ ▪▪▪

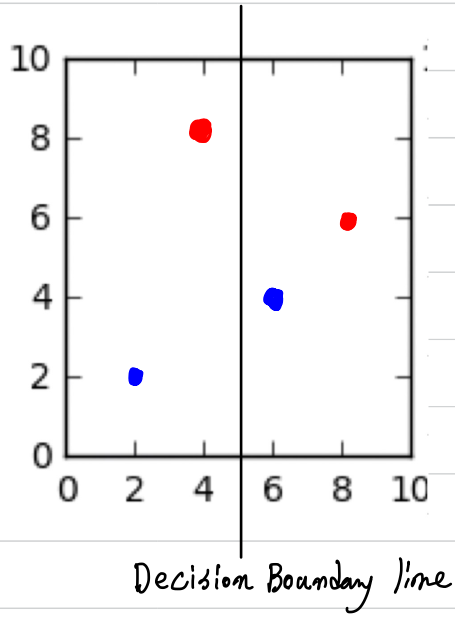
HW3 Part 2

1. $T(a + b \cdot x_1)$

- peceptron
- Two parameter and one feature (b)
- This learner can shattered dataset (a) &(b).
- This learner can't shattered dataset (c) &(D), if point $h \geq 3$

example c)



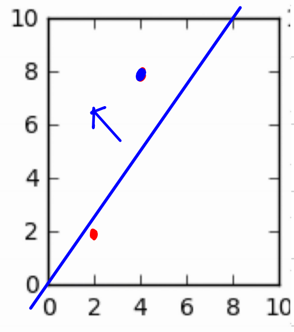


VCdim = 2 : Can arrange two points, can't arrange more than two

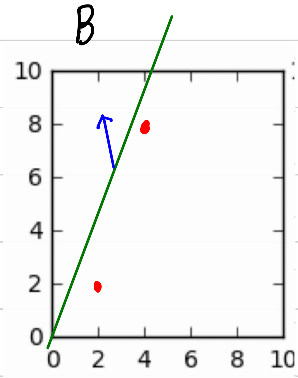
2. $T((a*b)x_1 + (c/a)\ x_2)$

- peceptron
- Three parameter(a,b,c) and two feature (a*b) & (c/a)
- No constant → Pass the orignal point (0, 0)
- This learner can shattered dataset (a) &(b).
- This learner can't shattered dataset (c) &(D), if point $n \geq 3$

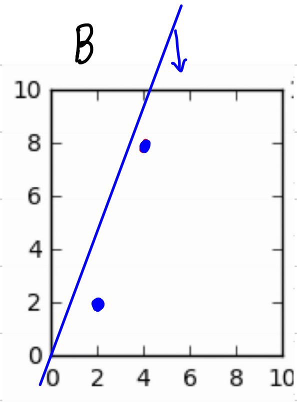
B : shattered



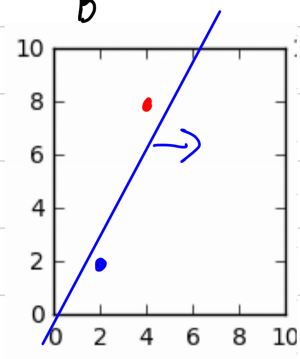
B



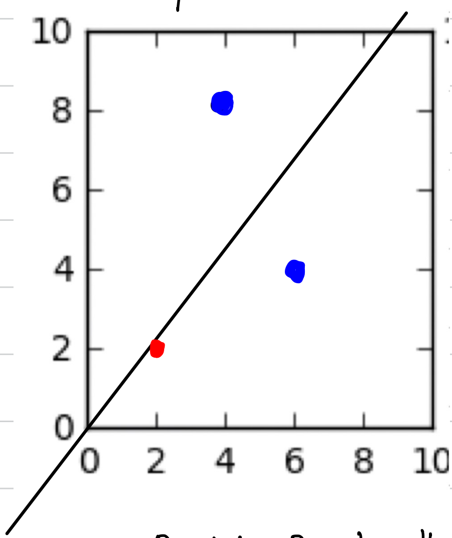
B



B

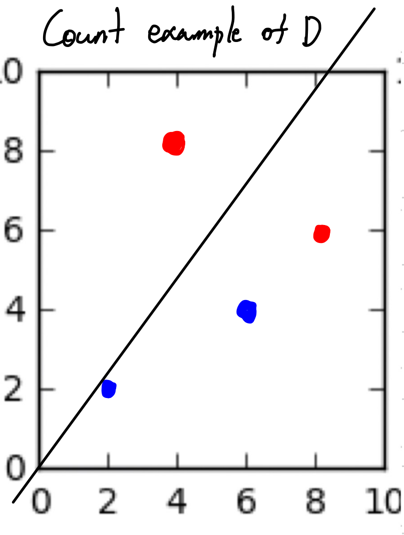


count example of D



Decision Boundary line.

Count example of D



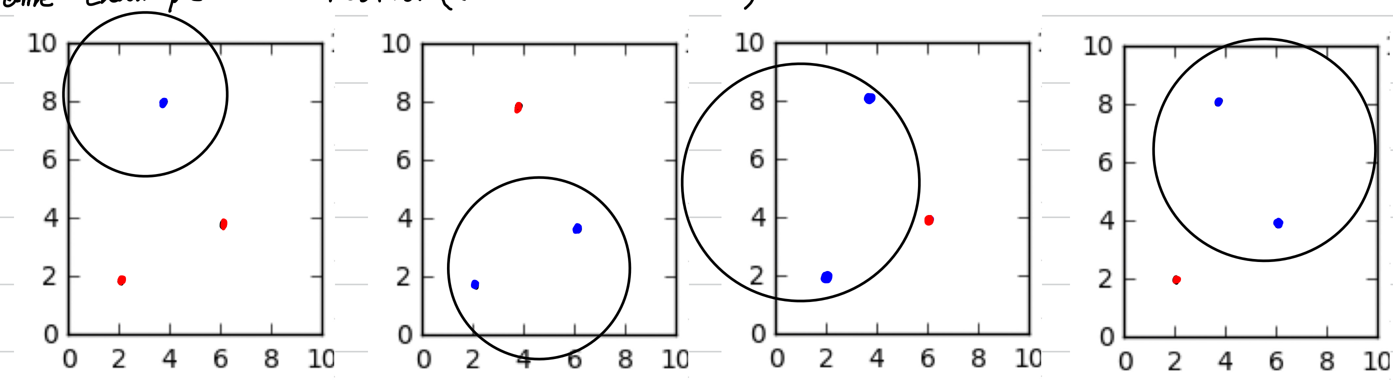
Decision Boundary line

VCdim = 2 : Can arrange two points, can't arrange more than two

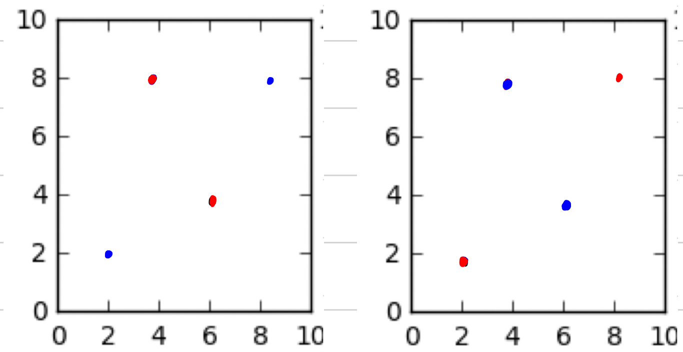
3. $T\left((x_1-a)^2+(x_2-b)^2+c\right)$

- three parameter, two features, one constant
- Circle type → center (a, b)

- Some example C : Learner $\left((x_1-a)^2+(x_2-b)^2+c\right)$ can shatter dataset C.



- Example this learner can't shatter dataset D



→ These two examples show this learner can't shatter data set D.

VCdim = 2 : Can arrange 2 points, can't arrange more than 3.

4. $T(a+bx_1+cx_2) \times T(d+bx_1+cx_2)$ VC Dimension = 3

- Total 4 parameters (a, b, c, d) and two features (b, c)
- there are two different constents (a) and (d).
- (a), (b), and (c) are shattered all of them
  - one more line than (b) so VC demension should be bigger than 2.

- this learner can shatter (d)