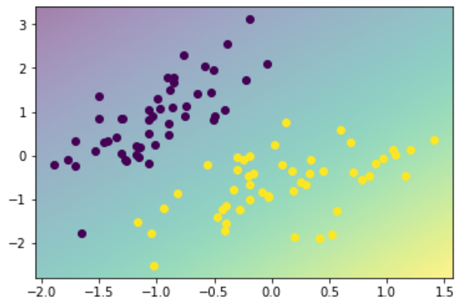
```python
import mltools as ml
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(0)
from logisticClassify2 import *
```

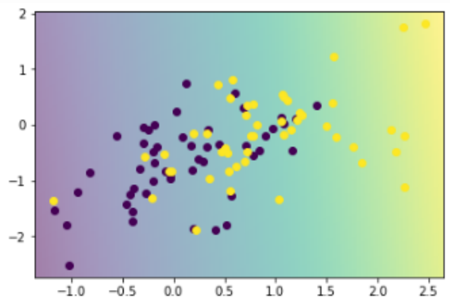
```python
iris = np.genfromtxt("data/iris.txt",delimiter=None)
X, Y = iris[:,0:2], iris[:,-1] # get first two features & target
X,Y = ml.shuffleData(X,Y) # reorder randomly (important later)
X,_ = ml.transforms.rescale(X) # works much better on rescaled data
XA, YA = X[Y<2,:], Y[Y<2] # get class 0 vs 1
XB, YB = X[Y>0,:], Y[Y>0] # get class 1 vs 2
```

Problem1A

```python
#learner = logisticClassify2()
lr = ml.linear.linearRegress(XA,YA)
ml.plotClassify2D(lr,XA,YA)
#plt.plot (XA,YA,'r.',XB,YB,'g.')
plt.show()
```



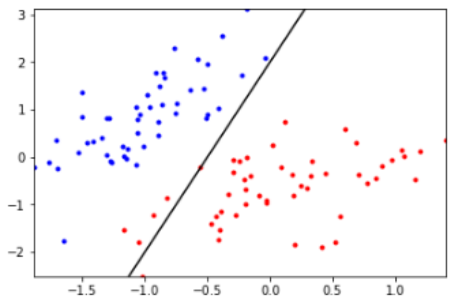
```python
#learner = logisticClassify2()
lr = ml.linear.linearRegress(XB,YB)
ml.plotClassify2D(lr,XB,YB)
#plt.plot (XA,YA,'r.',XB,YB,'g.')
plt.show()
```



Problem 1B

DATA A

```
]: learner = logisticClassify2(); # create "blank" learner
   learner.classes = np.unique(YA)
   #learner.theta = ([.5,1.-.25])
   theta0 = .5
   theta1 = 1
   theta2 = -.25
   wts = np.array([theta0,theta1,theta2]); # TODO: fill in values
   learner.theta = wts; # set the learner's parameters
   logisticClassify2.plotBoundary(learner,XA,YA)
   plt.show()
```

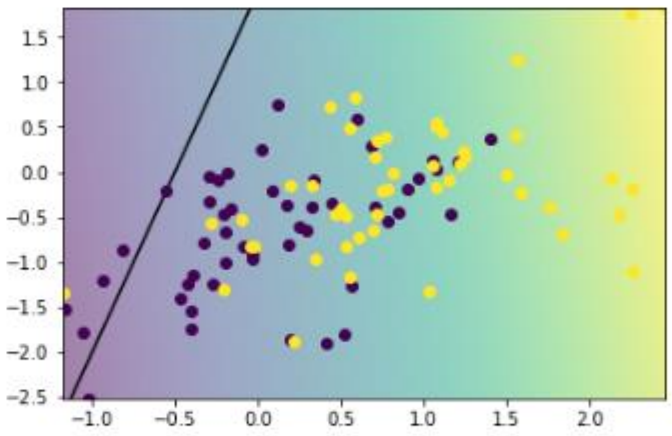


DATA B

```
learner2 = logisticClassify2(); # create "blank" learner
learner2.classes = np.unique(YB)
learner2.theta = ([.5,1,-.25])

logisticClassify2.plotBoundary(learner2,XB,YB)
ml.plotClassify2D(lr,XB,YB)
plt.show()
```

```
3
```



For some reason when i try plotting data B, It only plots in one

#color.However, for Data A everything works fine. But to show the

#different classes for DATA B i graph the data using

#ml.plotClassify2D(lr,XB,YB)

def plotBoundary(self,X,Y):

```
        print(len(self.theta))
        """ Plot the (linear) decision boundary of the classifier, along with data """
        #if len(self.theta != 3):
           # raise ValueError('Data & model must be 2D');
        #print("hello world")
        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 + theta1 X1 + theta2 X2 == 0
        x1b = np.array([ax[0],ax[1]]);  # at X1 = points in x1b
        x2b = -self.theta[0]/self.theta[2] - self.theta[1]/self.theta[2]*x1b    # TODO find x2 values as a function of x1's v
    alues
        ## Now plot the data and the resulting boundary:
        A = Y==0;                                              # and plot it:
        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 1C

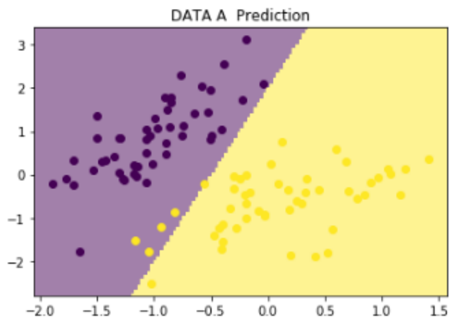
Implementation of Predict Function

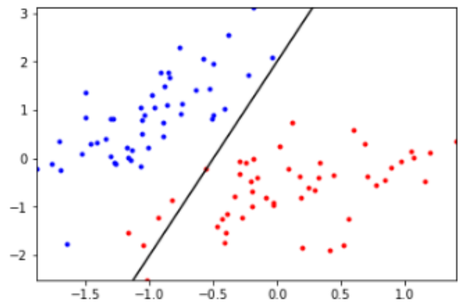
def predict(self, X):

```
""" Return the predictied class of each data point in X"""
#print("Hello world")
#print(X)
Yhat = []
r = []
for i in range(0,len(X)):
    r.append(self.theta[0] + self.theta[1]*X[i,0] + self.theta[2] * X[i,1])
    if r[i]>0:
        Yhat.append(self.classes[1])
    else:
        Yhat.append(self.classes[0])
#Yhat = self.classes(1+ ((self.theta[0] + X * self.theta[1:])))
## TODO: compute linear response r[i] = theta0 + theta1 X[i,1] + theta2 X[i,2] + ... for each i
## TODO: if z[i] > 0, predict class 1:  Yhat[i] = self.classes[1]
##        else predict class 0:  Yhat[i] = self.classes[0]
return np.array(Yhat);
```

Problem 1D

```
learner = logisticClassify2(); # create "blank" learner
learner.classes = np.unique(YA)
learner.theta = ([.5,1,-.25])
#Data A
#learner.plotboundary(XA, YA)
learner.predict(XA)
plt.title("DATA A  Prediction")
ml.plotClassify2D(learner,XA,YA)
plt.show()
logisticClassify2.plotBoundary(learner,XA,YA)
plt.show()
```

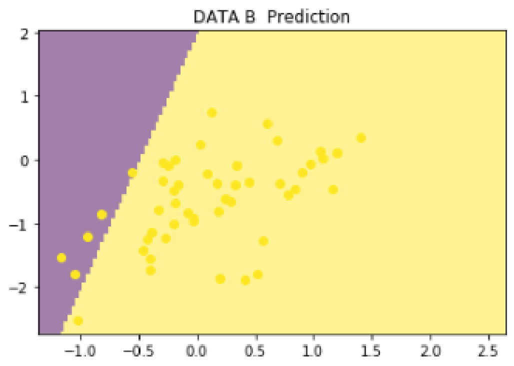




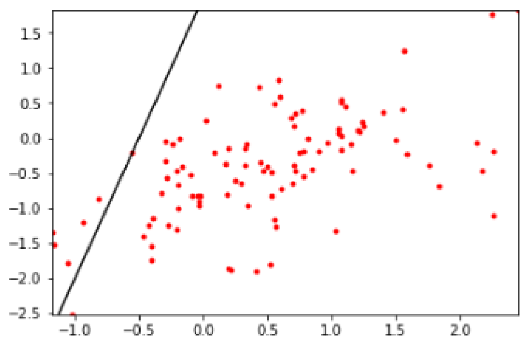
```
#Data B
#learner.plotboundary(XB, YB)
learner.predict(XB)
plt.title("DATA B  Prediction")


ml.plotClassify2D(learner,XB,YB)

plt.show()
logisticClassify2.plotBoundary(learner,XB,YB)
plt.show()
```



3



Problem 2 ¶

PART A

T(a+bx_1) is a learner that is composed of only one feature x_1, which also has a VC dimension 2. The decision boundtries of this learner are streight lines, and with each side of the line being able to get a negative or positive class value. Therefore, this learner can shatter graph (a) and (b) only. This is because of how the data is detributed.

PART B

The learner has three parameters and is composed of two features; x_1 and x_2. Also, the decision boundy of this learner will be a circle which devides the negative class (inside the circle) and the positive class (outside the circle). The reason why the positive class is located outside is because the values for a and b increase in distance. Since this is a circle, the radius is composed of the C constant. Now, that we know that our decision boundry is a circle then we can see that this learner can shatter graphs (a) (b) and (c). The reason why this learner can shatter graph (b) is because if one point is negative then we can make the circle around it and then the other will automatically will be consider positive. For graph (c), if we got two points that are negative then we set the decision boundry around those two points.

Part C

The learner for part c has two features; x_1 and x_2 and three parameters a,b,c which are not independent of each other because of the multiplication and devision operation. Also, we have to keep in mind that there wont be any independent constant. If we look at the learner function all the parameters are coefficients next to the features, therefore, the decision boundry will aways cross the orign an x-y axis. So with that in mind, we now know that graph (b) can be shattered because if one point is neg and the other point is pos , then a line could separate the classes. In the other hand, graph (c) cant be shattered because there if the right most point were to be both negative then, since the line has to pass through the origin, then there would be no way to separate them. Same interpretation goes for graph (d)