Assignment2

Shravan Kumar P

February 24, 2017

Here I reported my assignment work with respect to the questions given. All the code is implemented using python in Jupyter Notebook.

Contents

1 Problem 1

1.1 Perceptron Algorithm:

1.Use the data given in Table 1. Starting with w = [0 0] T and b = 0,apply your program to learn a linear classifier to separate classes C 1 and C 2 . Plot the data points for C 1 and C 2 . Plot the final classifier learnt at the convergence. Note the number of iterations required for convergence.

The algorithm implemented as taught in class(referring R.Duda Text Book). The Class1 and class2 data has 10 instances and 2 features. similarly for C2and C3. I have agumented the 10×2 matrix with all ones in the beginning to make it a 10×3 matrix. and added bias(threshold) term into the weight vector. Therefore my new weight vector is of size 3×1 istead 2×1.

Along with I have multiplied Class2 samples with -1, so that now I can decide the samples as misclassified if they are less than zero.

If the particular sample get misclassified then we do weight update using gradient descent method.

samples	C_1		C_2		C_3	
	x_1	x_2	x_1	x_2	x_1	x_2
1	0.1	1.1	7.1	4.2	-3.0	-2.9
2	6.8	7.1	-1.4	-4.3	0.5	8.7
3	-3.5	-4.1	4.5	0.0	2.9	2.1
4	2.0	2.7	6.3	1.6	-0.1	5.2
5	4.1	2.8	4.2	1.9	-4.0	2.2
6	3.1	5.0	1.4	-3.2	-1.3	3.7
7	-0.8	-1.3	2.4	-4.0	-3.4	6.2
8	0.9	1.2	2.5	-6.1	-4.1	3.4
9	5.0	6.4	8.4	3.7	-5.1	1.6
10	3.9	4.0	4.1	-2.2	1.9	5.1

Table 1: Data for Problem 1

Figure 1: Table1

Listing 1: Perceptron

```
def percp(train, weights, nb_epoch=1):
    w = weights
    X = train
    for epoc in range(nb_epoch):
        print("************************")
        print("Epoch {}".format(epoc))
        print("************************")
        print(" ")
        print("current weights {}".format(w))
        print(" ")
        for i in range(len(X)):
            print("iteration{} on X{}".format(i,i))
            y = np.dot(X[i],w)
            count = 0
            if y<=0:
                print("xxxxxxx")
                print("Misclassified sample x{}".format(i))
                print("-------")
                w = w+X[i]
                print("-------")
                print("updated weight w = {}".format(w))
                print(":) :) :) ")
#                count +=1
#                print("inside {}".format(count))
#            print("outside {}".format(count))
        print(" ")
        print("Finally the best weight vector is {} at iteration {}".format(w,epoc))
        print(" ")
    print(w)
    return w
```

Linear Classfier that discrimnate the two classes C1, C2 using the above solution vector At Iteration 7, the algorithm find the solution vector as w = [13. -10.2 11.3]

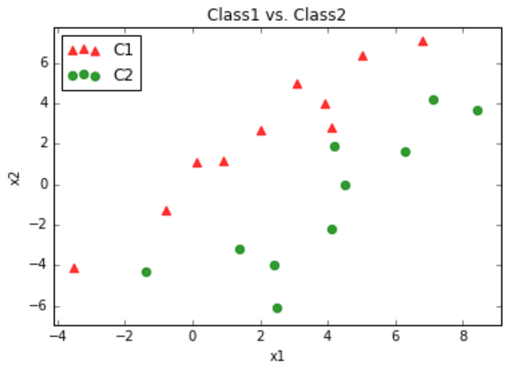


Figure 2: Class1 vs. Class2

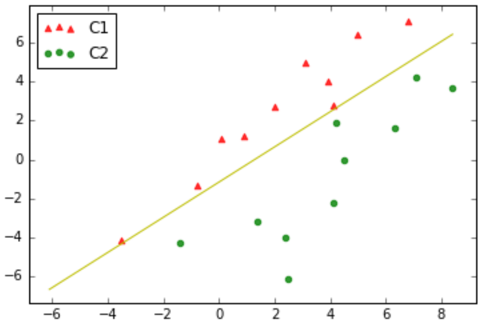


Figure 3: Linear classification

Linear Classfier that discrimnate the two classes C2, C3 using the above solution vector At Iteration 4, the algorithm find the solution vector as w = [-5. 5.5 -6.4]

From the figure above, the features of C1, C2 (figure (2)) are closely coupled as compared to the features of C2,C3 (figure(4)). Hence finding the optimally best solution vector for (C1,C2) takes more number of iterations thatn finding an optimally best solution for (C2,C3), 7, 4 number of iterations respectively.

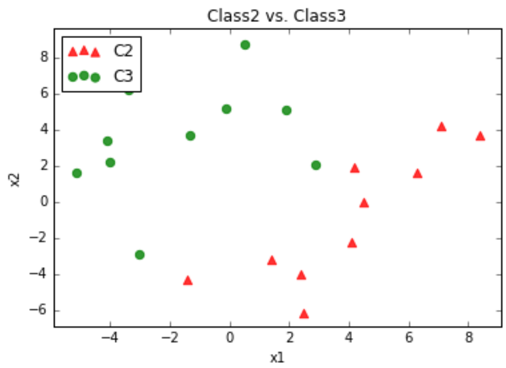


Figure 4: Class2 vs. Class3

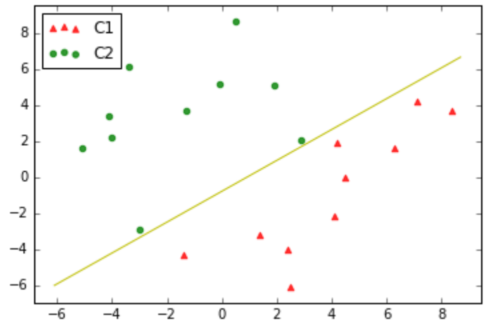


Figure 5: Linear Classification

2 Problem 2

voted-perceptron algorithm

Listing 2: Voted-Perceptron

```
def vpercp(X, Y, nb_epoch=1,kfold=10):
    w1 = np.zeros(features.shape[1])
    c1 = 1
    v1 = []
    p1 = []
    for epoch in range(nb_epoch):
        for x,y in zip(X,Y):
            u = np.inner(x,w1)
            if y*u<=0:
                v1.append(w1)
                p1.append(c1)
                w1 = w1+y*x
                c1 = 1
            else:
                c1 =c1+1
```

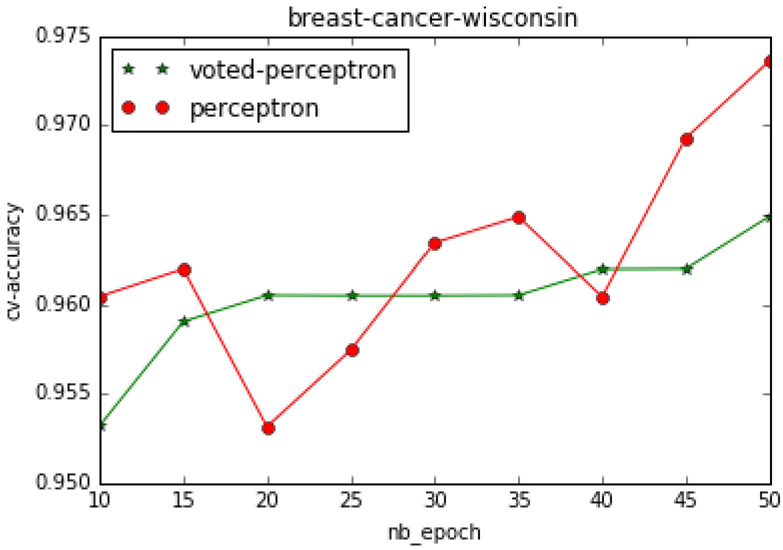
```
    return v1,p1
```

perceptron algorithm

Listing 3: Perceptron

```
def percp(X, Y, nb_epoch=1):
    w = np.zeros(X.shape[1])
    for epoch in range(nb_epoch):
        for x,y in zip(X,Y):
            if y*np.dot(x,w)<=0:
                w = w+y*x
    return w
```

2.1 breast-cancer-wisconsin

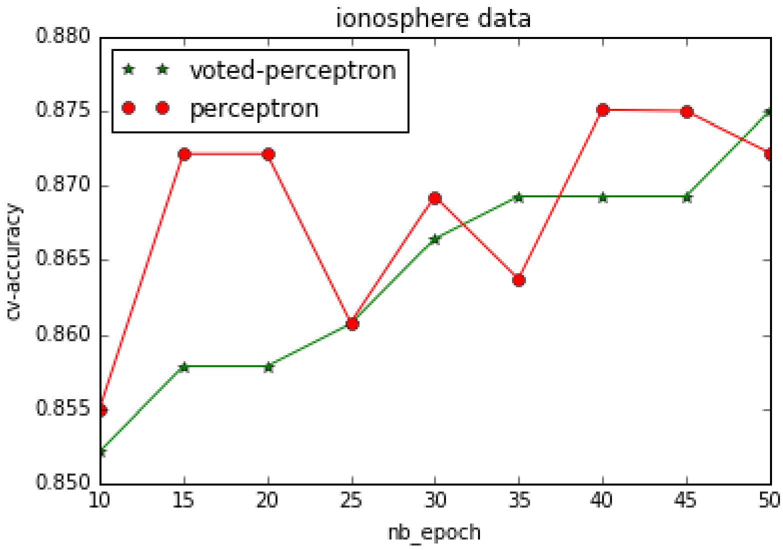


Remove the rows corresponding to missing values in Breast Cancer Dataset

Listing 4: To remove missing values

```
df = df[(df[6]!='?')].astype('float')
del df[0]
n = 1
```

2.2 ionosphere


ionosphere data
voted-perceptron
perceptron
cv-accuracy
nb_epoch
0.880
0.875
0.870
0.865
0.860
0.855
0.850
10
15
20
25
30
35
40
45
50

3 Problem 3

Figure 6

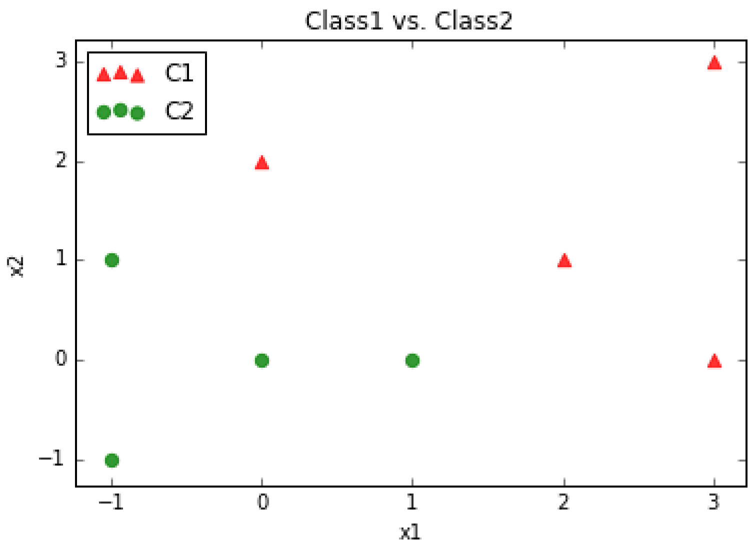


Figure 6: C1vsC2-Table1

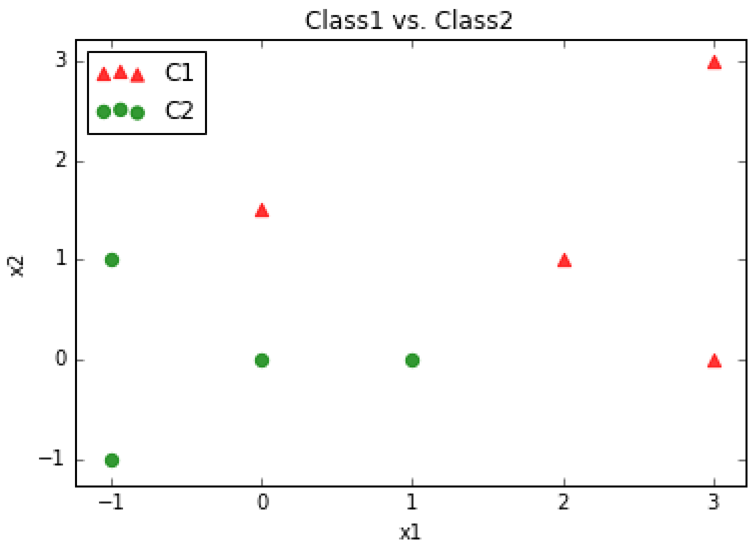


Figure 7: C1vsC2-Table2

3.1 Least Squares Method

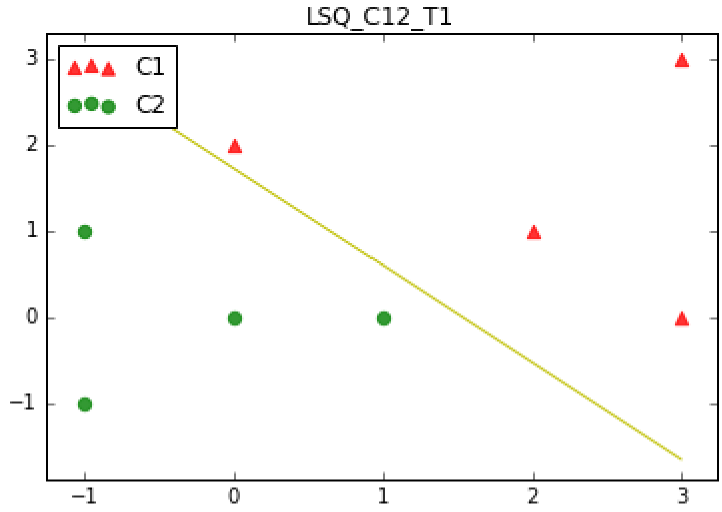


Figure 8: LSQ-C1vsC2-Table1

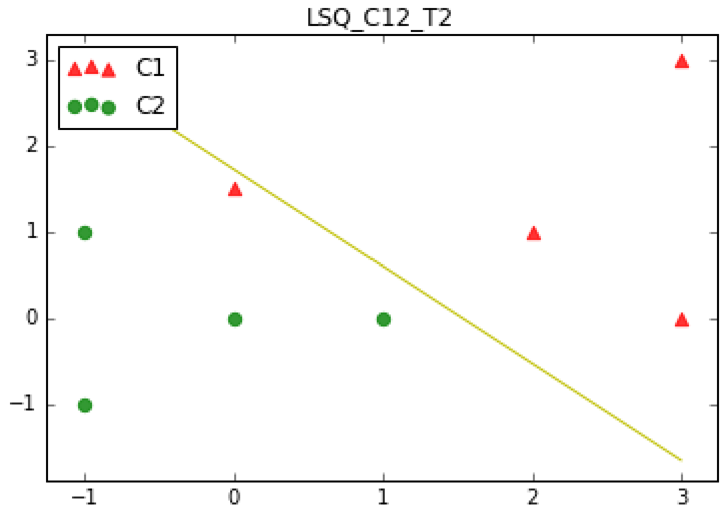


Figure 9: LSQ-C1vsC2-Table2

Listing 5: Least Square Method

```
def lsq(X12,b=None):
    b = np.ones(len(X12))
    w = np.dot(np.linalg.pinv(X12),b)
    return w
```

3.2 Fisher's LDA

Listing 6: Fisher's Linear Discriminant

```
def flda(c1,c2,p=False):
    u1 = np.mean(c1, axis=0)
    u2 = np.mean(c2, axis=0)
    S1 = (len(c1) -1)*np.cov(c1.T)
    S2 = (len(c2 -1))*np.cov(c2.T)
    Sw = np.add(S1,S2)
    iSw = np.linalg.pinv(Sw)
    v = np.dot(iSw,(u1-u2))
    if p:
        v1 = np.c_[v, v]
        plotv(v1)
    return v
```

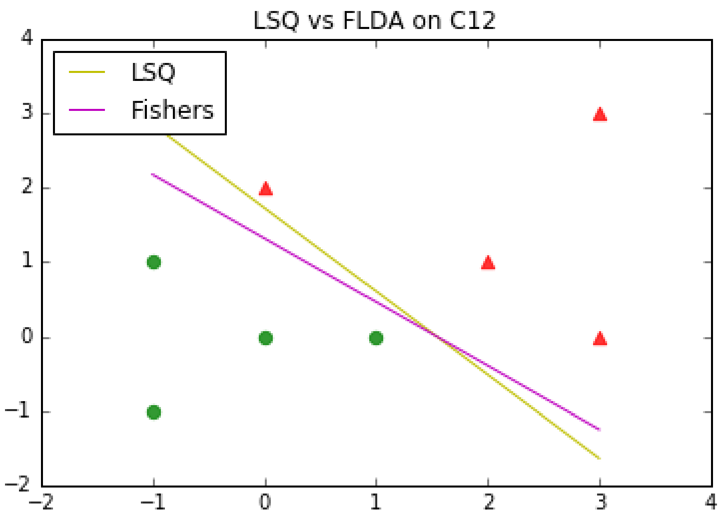


Figure 10: LSQ-FLDA-C1vsC2-Table1

4 Problem 4

4.1 Relation between Least Squares and Fisher's linear discriminant

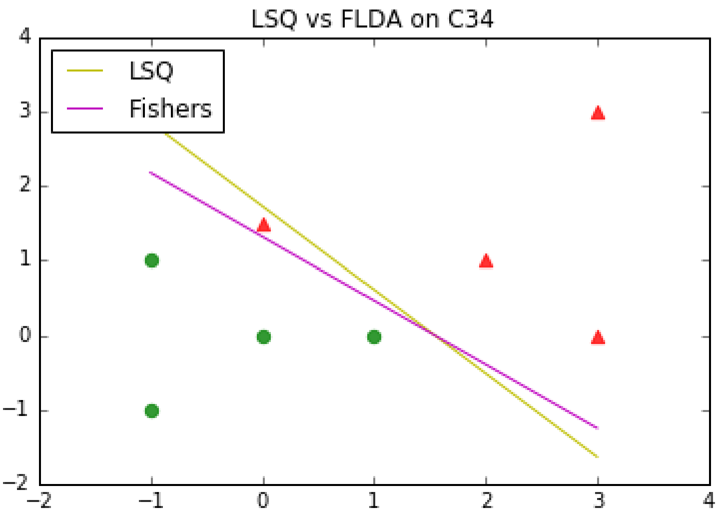


Figure 11: LSQ-FLDA-C3vsC4-Table2

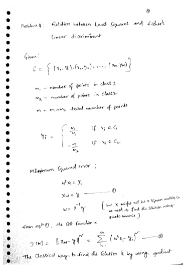


Figure 12: p1

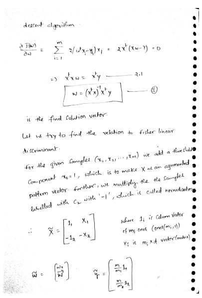


Figure 13: p2

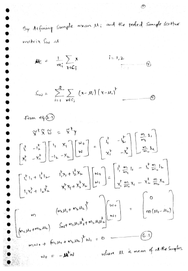


Figure 14: p3

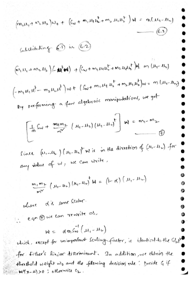


Figure 15: p4