ECE 532 Activity 19

Ayan Deep Hazra

1. $x_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ $x_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ $d_1 = -1, d_2 = 1$

Thus, $X = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$ $d = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ $w = \begin{bmatrix} -1 \\ 0.5 \end{bmatrix}$

a) Squared Error loss, thus,

Squared error loss

$$l(w_i; X, d) = \| Xw - d \|_2^2$$

$$= \left\| \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 0.5 \end{bmatrix} - \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right\|_2^2$$

$$= \left\| \begin{bmatrix} -0.5 \\ 1.5 \end{bmatrix} - \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right\|_2^2$$

$$= \left\| \begin{bmatrix} +0.5 \\ 0.5 \end{bmatrix} \right\|_2^2$$

$$= (+0.5)^2 + (+0.5)^2$$

$$= 0.25 + 0.25 = 0.5$$

Scanned with CamScanner

b) Hinge loss $= \sum_{i=1}^{N} (1 - d_i x_i^T w)_+$

$= (1 - d_1 x_1^T w)_+ + (1 - d_2 x_2^T w)_+$

$= \left(1 - (-1)\begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}-1 \\ 0.5\end{bmatrix}\right)_+ + \left(1 - (1)\begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}-1 \\ 0.5\end{bmatrix}\right)_+$

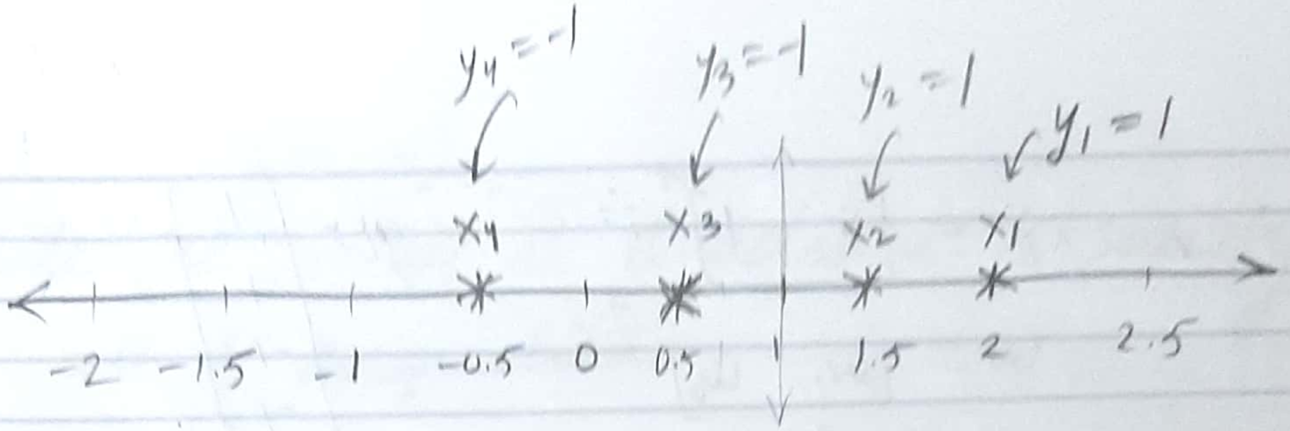
$= (1 - (-1)(-0.5))_+ + (1 - (1)(1.5))_+$

$= (1 - 0.5)_+ + (1 - 1.5)_+$

$= 0.5 + 0$

$= 0.5$

2.



a) The two closest points that are classified differently are x_2 & x_3.

Thus, the max margin classifier is the midpoint between the two on the real line.

Thus, max margin classifier $= \frac{x_2 + x_3}{2} =$

$$= \frac{0.5+1.5}{2} = \frac{2}{2} = 1$$

or $\boxed{x=1}$

b) This classifier does not make any errors (Check code later)

c) Zero hinge loss.

$$1 - y_i x_i^T w \leq 0$$

$$y_i x_i^T w \geq 1$$

$X_1 = [x_1 \quad 1]^T$ $\qquad y_i = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$

$W = [w_1 \quad w_0]^T$

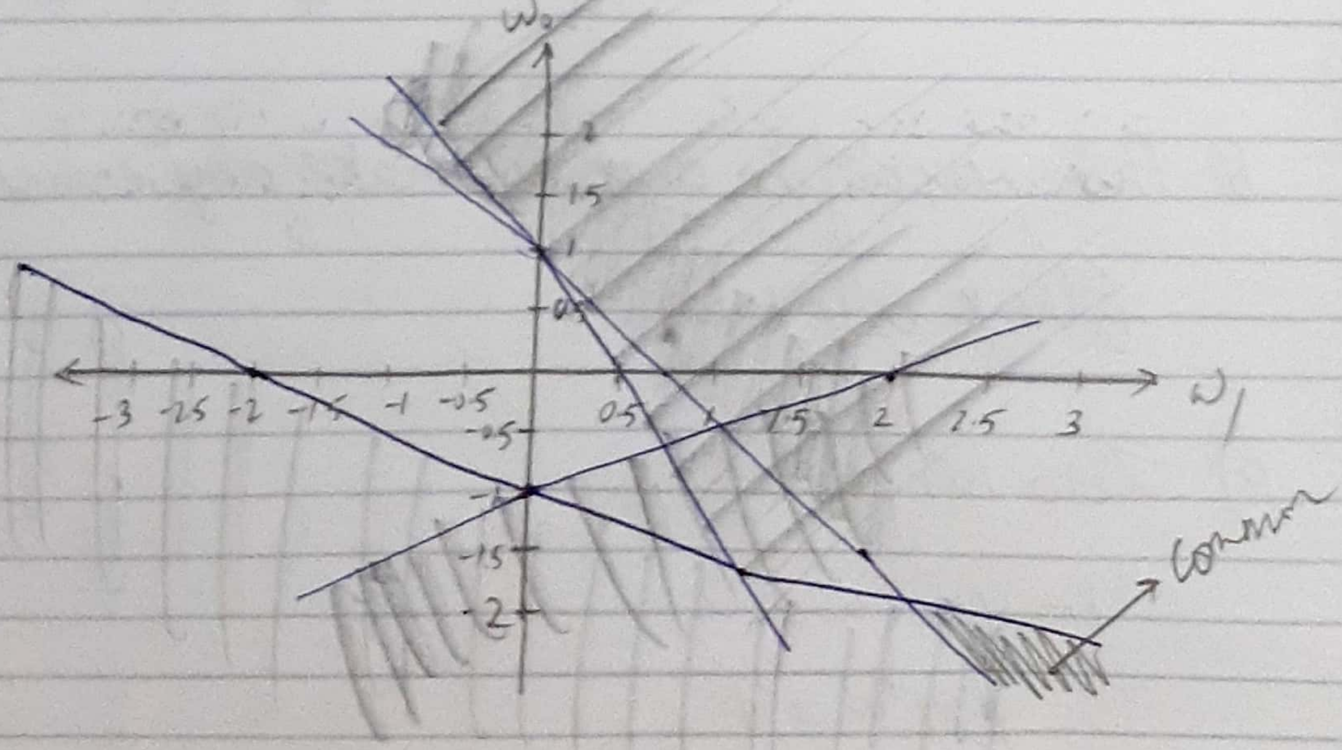
By doing $y_i x_i^T w_i$ we get a set of 4 inequalities

$1(2w_1 + w_0) \geq 1$

$1(1.5w_1 + w_0) \geq 1$

$-1(0.5w_1 + w_0) \geq 1$

$-1(-0.5w_1 + w_0) \geq 1$



We see (w_1, w_0) values in Quadrant 4 satisfy all 4 inequalities.

Ex. $(w_1 = 100, w_0 = -100)$

This classifier makes no errors.

d) This classifier makes errors.
(Find in code).

e) Yes, we can find a classifier with zero hinge loss when $x_4 = -5$.

→ It makes no errors.

The misclassified point gives 0 error when classified with Hinge loss.

3 a) There are 1213 classification errors with this svm.

b) There are 495 errors with this squared error classifier.

c) The svm is not affected at all by the newly added points and has the same number of errors as before.

(No change)

d) The error rate increases by a lot after the new points are added.

(nearly 2668 errors)

The svm proves to be much better when data points are far away from the decision boundary.

This is a result of the squared error classifier placing a lot of emphasis on the datapoints' distance from the boundary of separation.

Problem 2

b

In [1]:
```
import numpy as np
from scipy.io import loadmat
import matplotlib.pyplot as plt
from sklearn.svm import LinearSVC

X = np.array([[2,1],[1.5,1],[0.5,1],[-0.5,1]])
y = np.array([[1],[1],[-1],[-1]])

wLS = np.linalg.inv(X.T@X)@X.T@y
print("weight vector :")
print(wLS)

yout = np.sign(X@wLS)

print("y obtained from wLS :")
print(yout)
print("element matching of yout and y :")
print(yout==y)
```

```
weight vector :
[[ 0.94915254]
 [-0.83050847]]
y obtained from wLS :
[[ 1.]
 [ 1.]
 [-1.]
 [-1.]]
element matching of yout and y :
[[ True]
 [ True]
 [ True]
 [ True]]
```

d

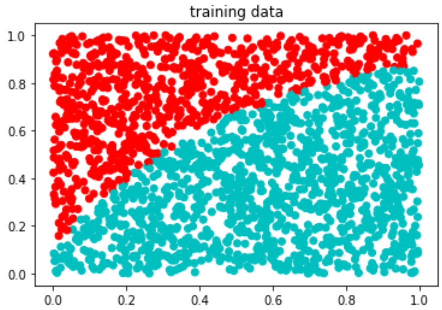
In [2]:
```

X = np.array([[2,1],[1.5,1],[0.5,1],[4,1]])
y = np.array([[1],[1],[-1],[-1]])

wLS = np.linalg.inv(X.T@X)@X.T@y
print("weight vector :")
print(wLS)

yout = np.sign(X@wLS)

print("y obtained from wLS :")
print(yout)
print("element matching of yout and y :")
print(yout==y)
```

```
weight vector :
[[-0.15384615]
 [ 0.30769231]]
y obtained from wLS :
[[ 0.]
 [ 1.]
 [ 1.]
 [-1.]]
element matching of yout and y :
[[False]
 [ True]
 [False]
 [ True]]
```

Problem 3

In [3]:

```
in_data = loadmat('classifier_data.mat')

x_train = in_data['x_train']
x_eval = in_data['x_eval']
y_train = in_data['y_train']
y_eval = in_data['y_eval']

n_eval = np.size(y_eval)
n_train = np.size(y_train)

plt.scatter(x_train[:,0],x_train[:,1], color=['c' if i==-1 else 'r' for i in y_train[:,0]])
plt.title('training data')
plt.show()
```

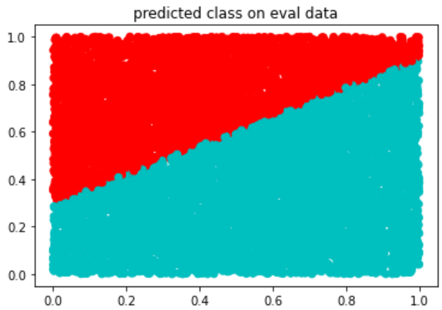


In [4]:

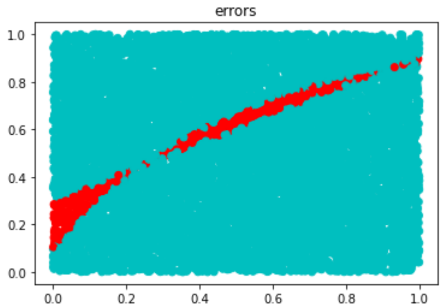
```
plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==-1 else 'r' for i in y_eval[:,0]])
plt.title('eval data true class')
plt.show()
```

In [5]:

```
## Classifier 1
x_train_1 = np.hstack(( x_train, np.ones((n_train,1)) ))
x_eval_1 = np.hstack(( x_eval, np.ones((n_eval,1)) ))

# Train classifier using linear SVM from SK Learn library
clf = LinearSVC(random_state=0, tol=1e-8)
clf.fit(x_train_1, np.squeeze(y_train))
w_opt = clf.coef_.transpose()

#uncomment this line to use least squares classifier
w_opt = np.linalg.inv(x_train_1.T@x_train_1)@x_train_1.T@y_train

y_hat_outlier = np.sign(x_eval_1@w_opt)
plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==-1 else 'r' for i in y_hat_outlier[:,0]])
plt.title('predicted class on eval data')
plt.show()
```



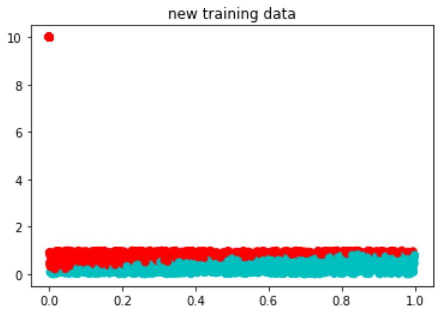
In [6]:
```python
error_vec = [0 if i[0]==i[1] else 1 for i in np.hstack((y_hat_outlier, y_eval))]
plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==0 else 'r' for i in error_vec])
plt.title('errors')
plt.show()

print('Errors: '+ str(sum(error_vec)))
```



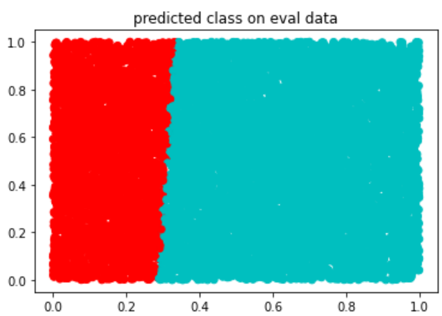

```
Errors: 495
```

Add correct points far from boundary

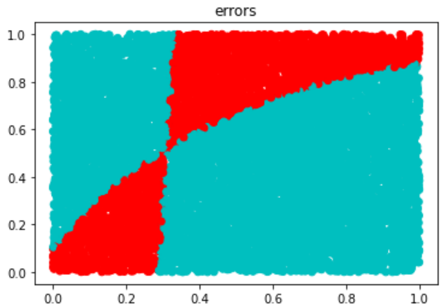
In [7]:
```python
## create new, correctly labeled points
n_new = 1000 #number of new datapoints
x_train_new = np.hstack((np.zeros((n_new,1)), 10*np.ones((n_new,1))))
y_train_new = np.ones((n_new,1))

## add these to the training data
x_train_outlier = np.vstack((x_train,x_train_new))
y_train_outlier = np.vstack((y_train,y_train_new))
plt.scatter(x_train_outlier[:,0],x_train_outlier[:,1], color=['c' if i==-1 else 'r' for i in y_train_outlier[:,0]])
plt.title('new training data')
plt.show()
```



In [8]:
```python
x_train_outlier_1 = np.hstack((x_train_outlier, np.ones((n_train+n_new,1)) ))
x_eval_1 = np.hstack((x_eval, np.ones((n_eval,1)) ))

#Train classifier using off the shelf SVM from sklearn
clf = LinearSVC(random_state=0, tol=1e-5)
clf.fit(x_train_outlier_1, np.squeeze(y_train_outlier))
w_opt_outlier = clf.coef_.transpose()

#uncomment this line to use least squares classifier
w_opt_outlier = np.linalg.inv(x_train_outlier_1.T@x_train_outlier_1)@x_train_outlier_1.T@y_train_outlier

y_hat_outlier = np.sign(x_eval_1@w_opt_outlier)
plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==-1 else 'r' for i in y_hat_outlier[:,0]])
plt.title('predicted class on eval data')
plt.show()
```



In [9]:
```
error_vec = [0 if i[0]==i[1] else 1 for i in np.hstack((y_hat_outlier, y_eval))]
plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==0 else 'r' for i in error_vec])
plt.title('errors')
plt.show()

print('Errors: '+ str(sum(error_vec)))
```




```
Errors: 2668
```

In [ ]: