

# linear\_classifier

February 16, 2023

1 2a)

```
[ ]: import numpy as np
from scipy.io import loadmat
import matplotlib.pyplot as plt

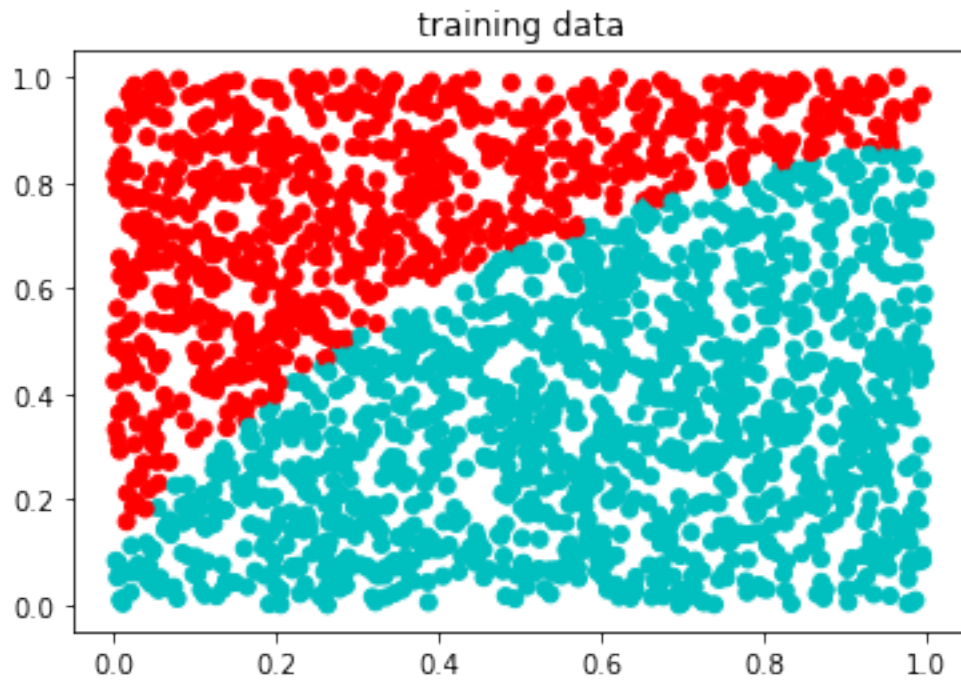
in_data = loadmat('classifier_data.mat')
print([key for key in in_data]) # -- use this line to see the keys in the
    ↪ dictionary data structure

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()
```

```
['__header__', '__version__', '__globals__', 'x_eval', 'x_train', 'y_eval',
'y_train']
```



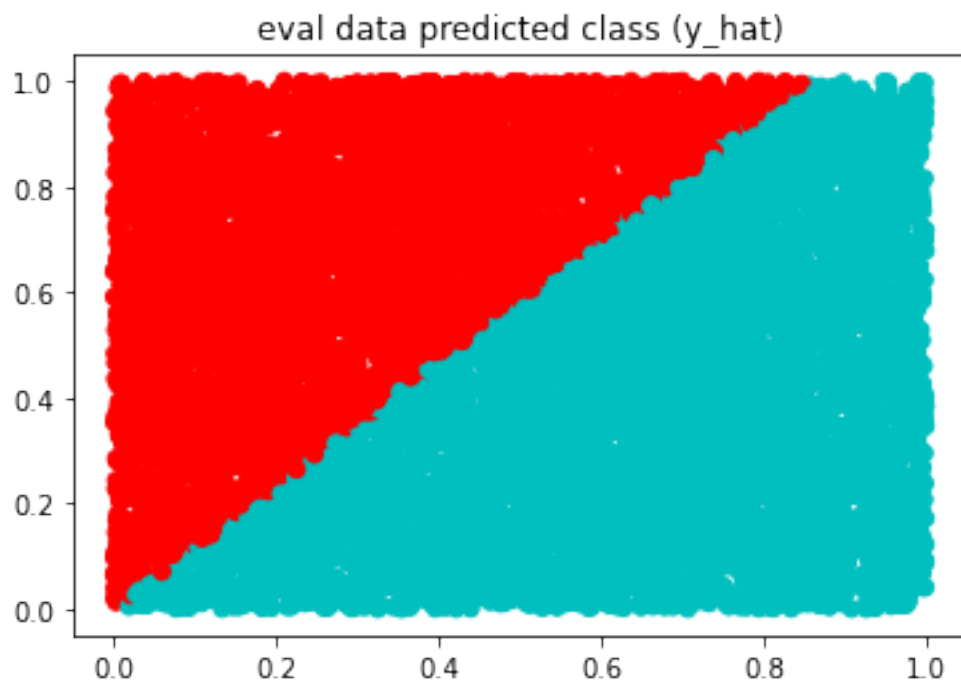
```
[ ]: plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==-1 else 'r' for i in
    ↪ y_eval[:,0]])
plt.title('eval data true class')
plt.show()
```



```
[ ]: ## Classifier 1

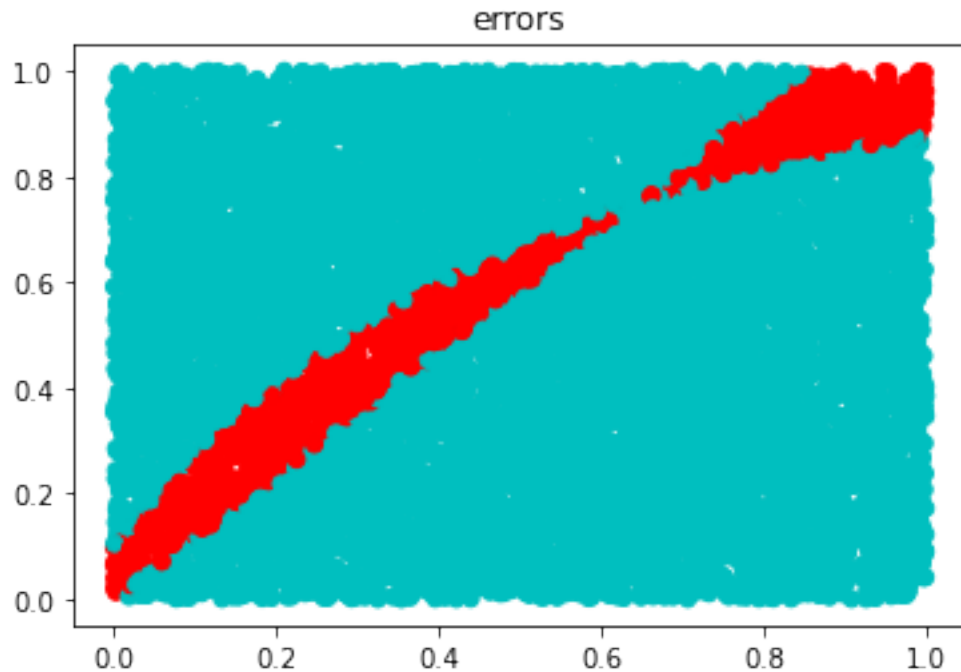
#  $w = (X^T X)^{-1} X^T y$ 
w_opt = np.linalg.inv(x_train.transpose()@x_train.transpose()@y_train
y_hat = np.sign(x_eval@w_opt)

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



```
[ ]: error_vec = [0 if i[0]==i[1] else 1 for i in np.hstack((y_hat, 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)))
```



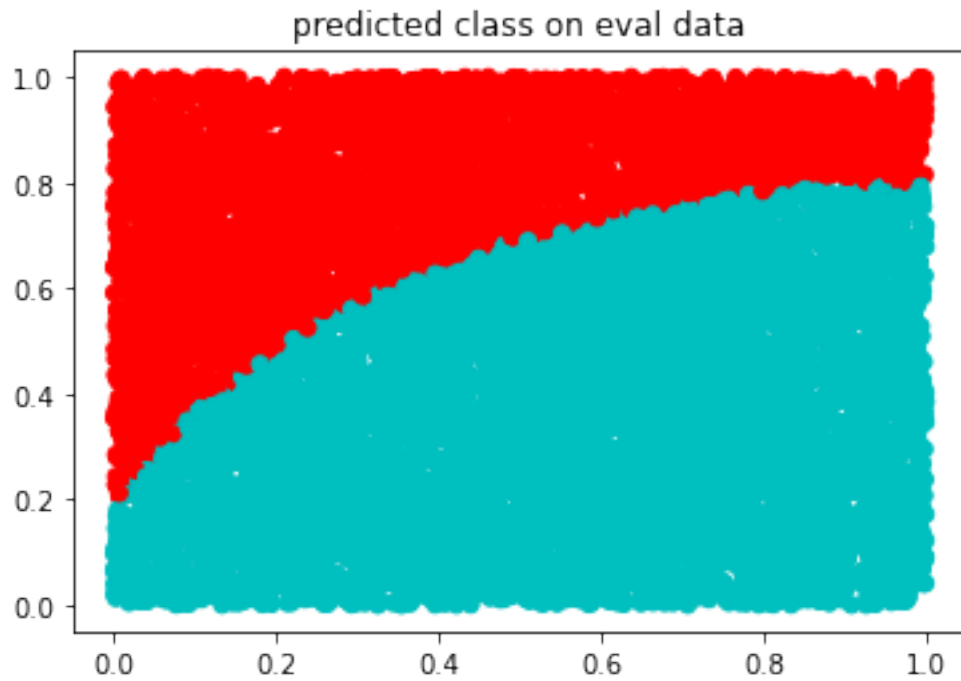
Errors: 1102

2 2b)

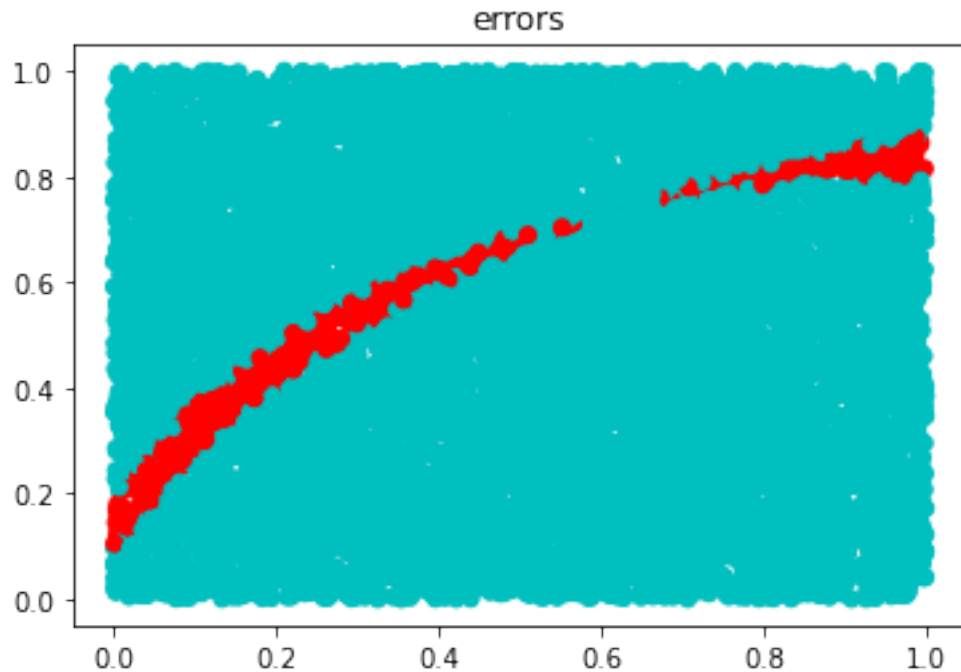
```
[ ]: ## Classifier 2
x_train_2 = np.hstack((x_train**2, x_train, np.ones((n_train,1)) ))
x_eval_2 = np.hstack((x_eval**2, x_eval, np.ones((n_eval,1)) ))

w_opt_2 = np.linalg.inv(x_train_2.transpose()@x_train_2.
    ↳transpose()@y_train
y_hat_2 = np.sign(x_eval_2@w_opt_2)

plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==-1 else 'r' for i in
    ↳y_hat_2[:,0]])
plt.title('predicted class on eval data')
plt.show()
```



```
[ ]: error_vec_2 = [0 if i[0]==i[1] else 1 for i in np.hstack((y_hat_2, y_eval))]  
plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==0 else 'r' for i in_  
    ↪error_vec_2])  
plt.title('errors')  
plt.show()  
  
print('Error: ' + str(sum(error_vec_2)))
```



Error: 542

### 3 2c)

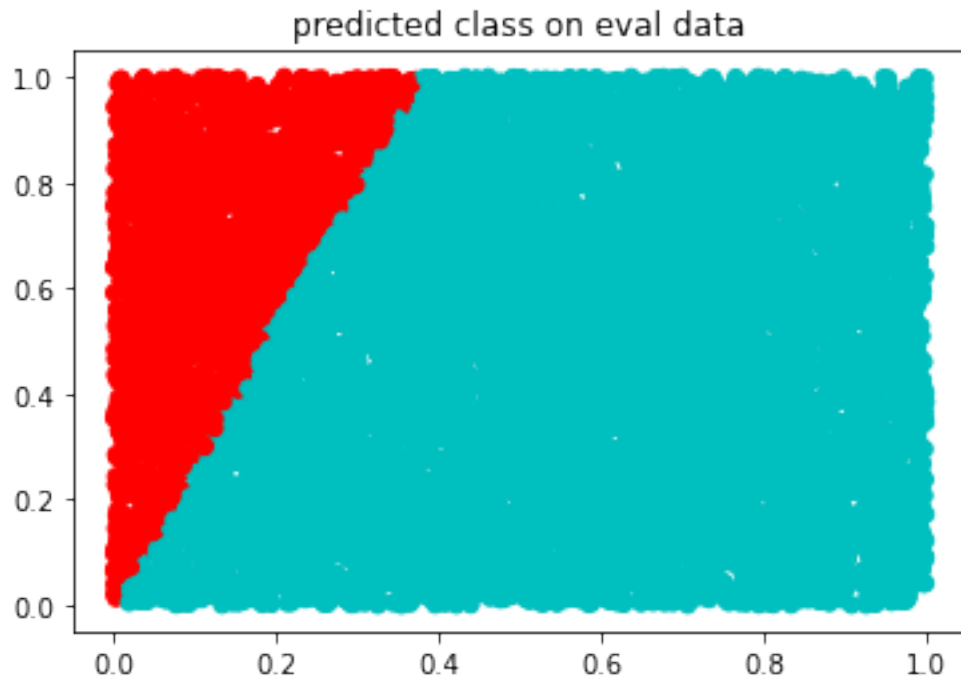
```
[ ]: ## create new, correctly labeled points
n_new = 1000 #number of new datapoints
x_train_new = np.hstack((np.zeros((n_new,1)), 3*np.ones((n_new,1))))
y_train_new = np.ones((n_new,1))

## 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()
```



```
[ ]: #train with new data
w_opt_outlier = np.linalg.inv(x_train_outlier.
    ↳ transpose()@x_train_outlier.transpose()@y_train_outlier
y_hat_outlier = np.sign(x_eval@w_opt_outlier)

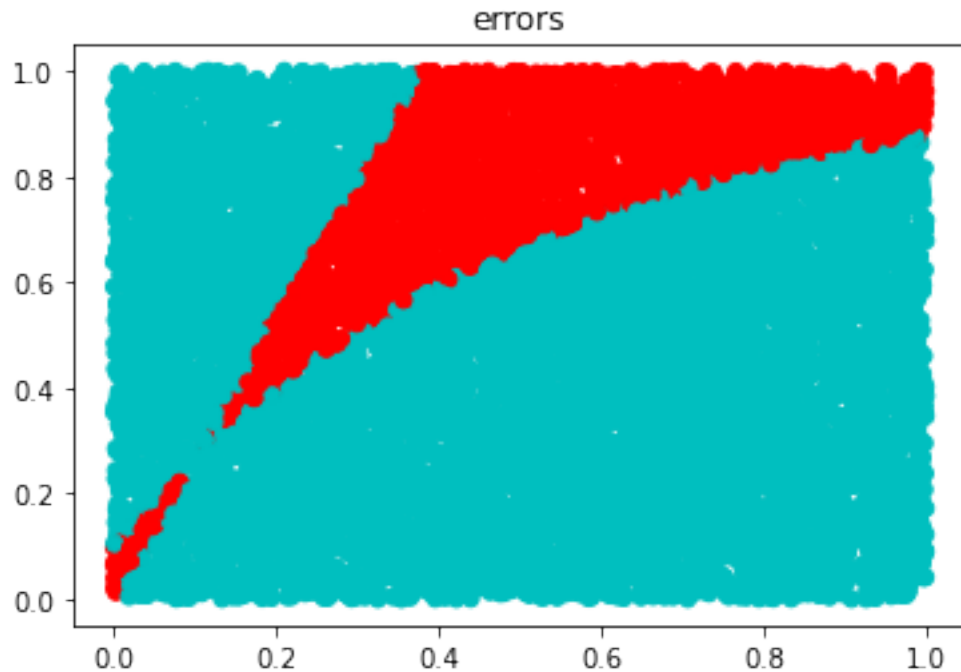
plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==-1 else 'r' for i in
    ↳ y_hat_outlier[:,0]])
plt.title('predicted class on eval data')
plt.show()
```



```
[ ]: 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: 2134

#### 4 3a)

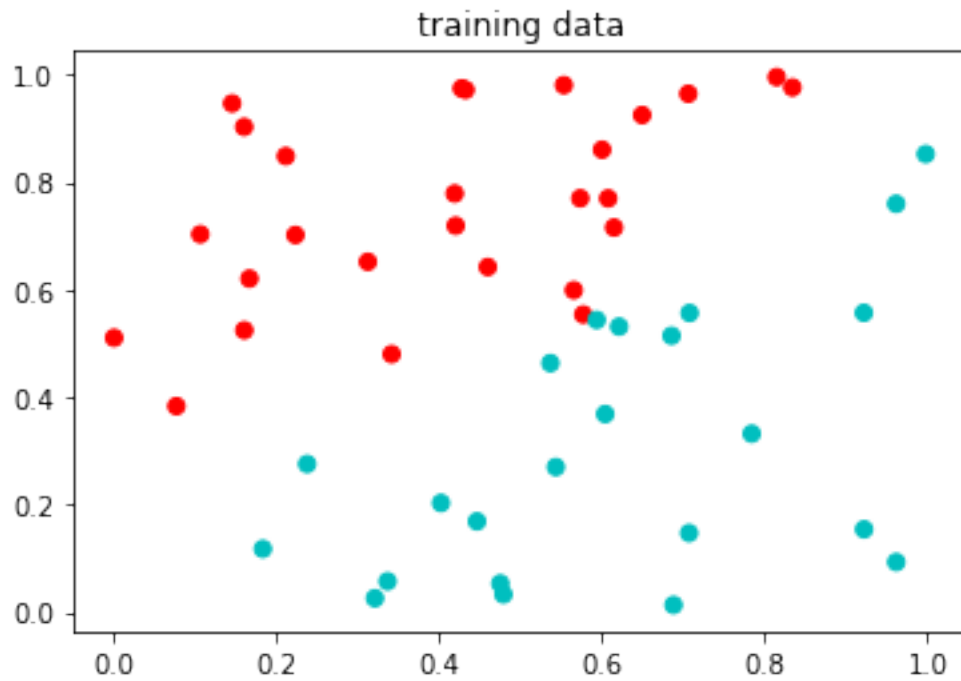
```
[ ]: in_data = loadmat('overfitting_data.mat')
print([key for key in in_data]) # -- use this line to see the keys in the
    dictionary data structure

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

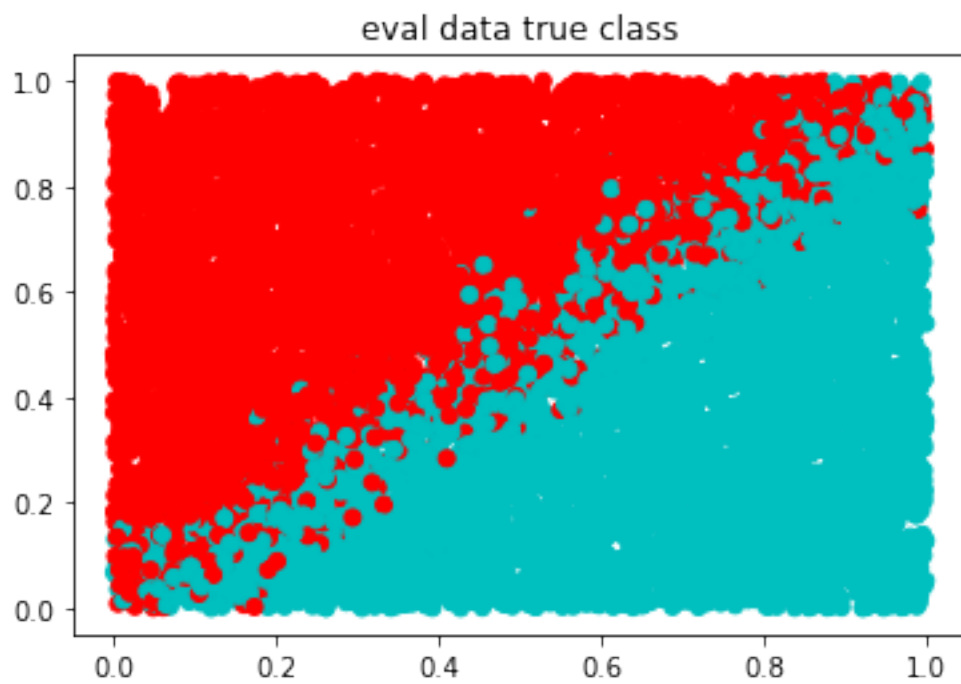
n_eval = np.size(y_eval)
n_train = np.size(y_train)

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()
```

```
['__header__', '__version__', '__globals__', 'x_eval', 'x_train', 'y_eval',
'y_train']
```



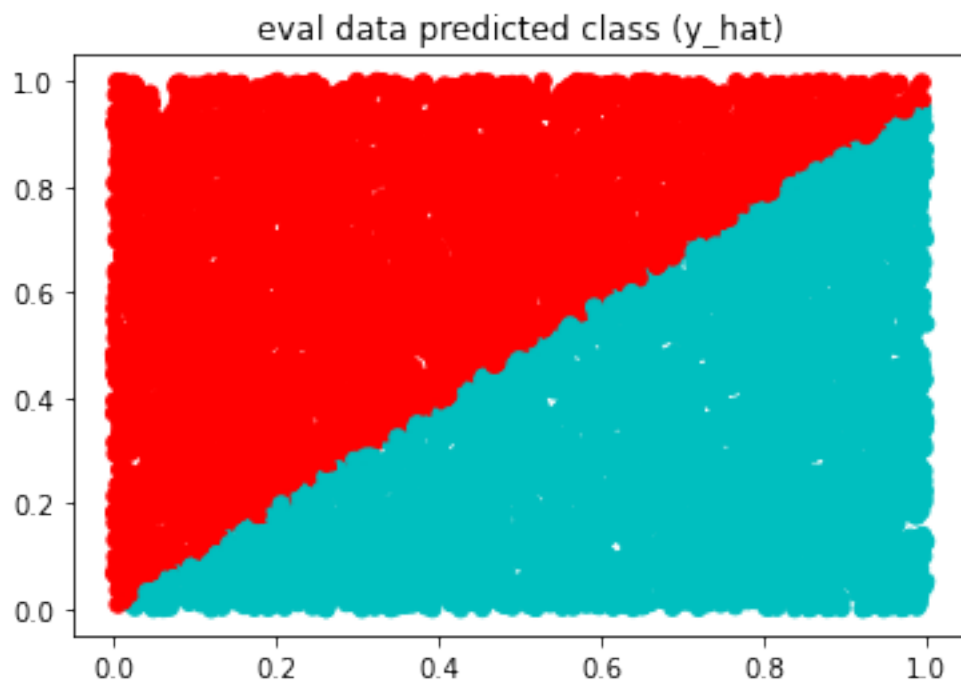
```
[ ]: plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==-1 else 'r' for i in y_eval[:,0]])  
plt.title('eval data true class')  
plt.show()
```



```
[ ]: ## Classifier 1

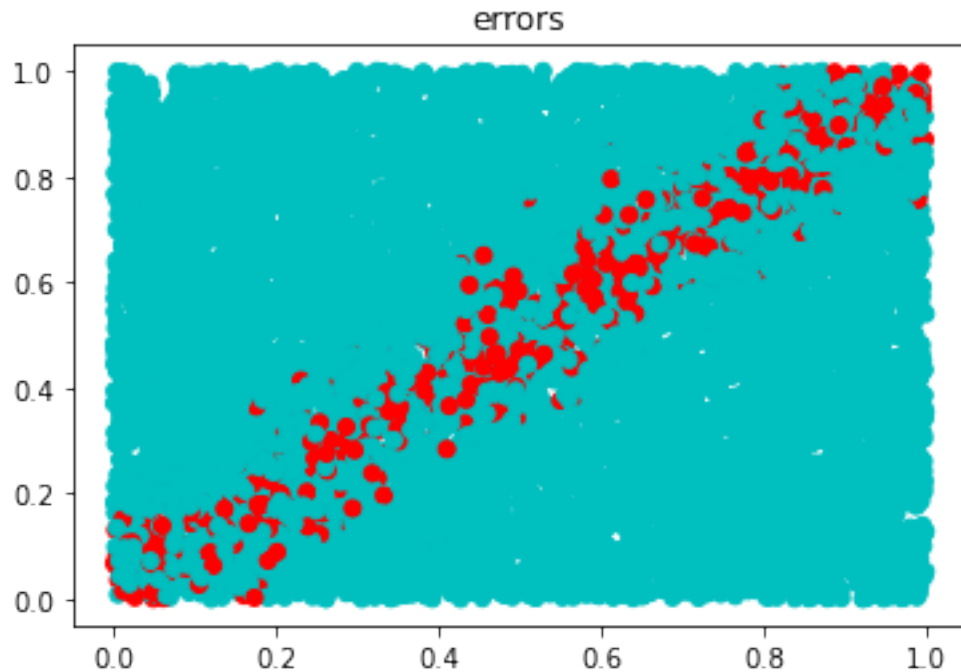
#  $w = (X^T X)^{-1} X^T y$ 
w_opt = np.linalg.inv(x_train.transpose()@x_train.transpose()@y_train
y_hat = np.sign(x_eval@w_opt)

plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==1 else 'r' for i in_
    ↪y_hat[:,0]])
plt.title('eval data predicted class (y_hat)')
plt.show()
```



```
[ ]: error_vec = [0 if i[0]==i[1] else 1 for i in np.hstack((y_hat, 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)))
```



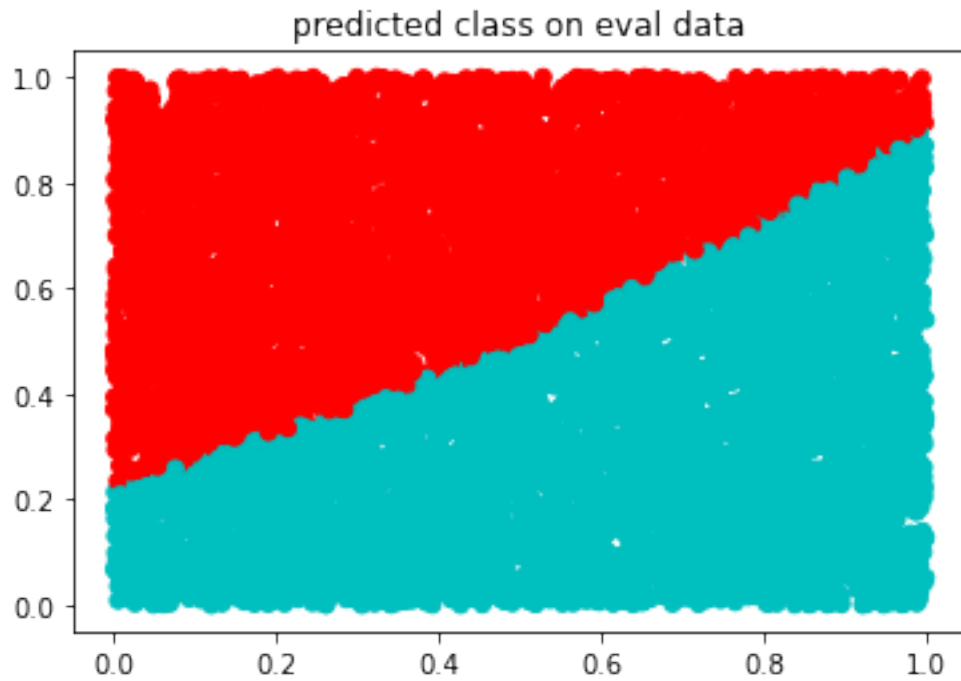
Errors: 759

## 5 3b)

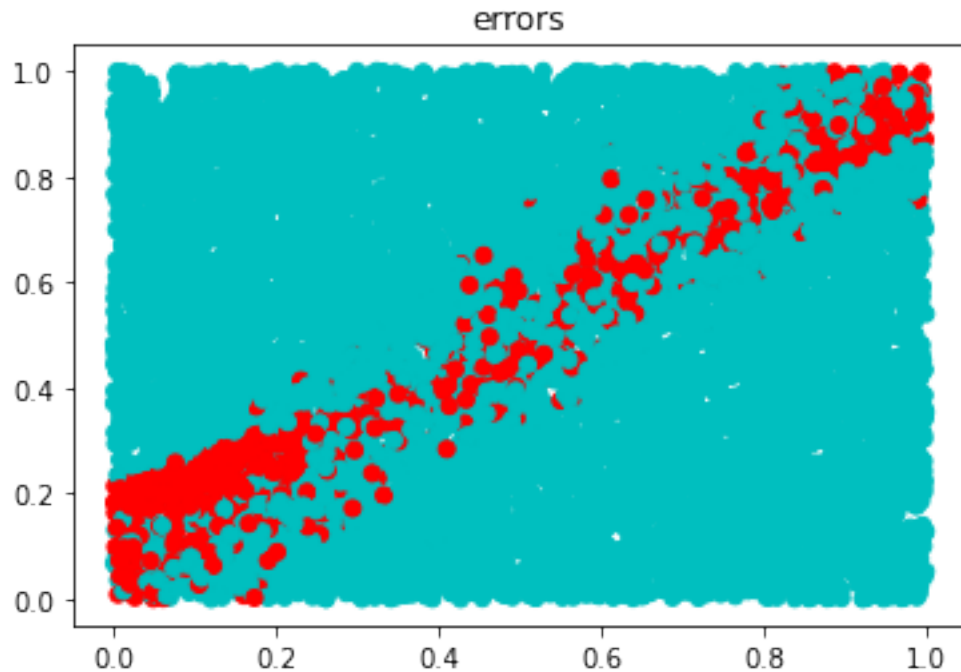
```
[ ]: ## Classifier 2
x_train_2 = np.hstack((x_train**2, x_train, np.ones((n_train,1)) ))
x_eval_2 = np.hstack((x_eval**2, x_eval, np.ones((n_eval,1)) ))

w_opt_2 = np.linalg.inv(x_train_2.transpose()@x_train_2).
    ↳transpose()@y_train
y_hat_2 = np.sign(x_eval_2@w_opt_2)

plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==-1 else 'r' for i in
    ↳y_hat_2[:,0]])
plt.title('predicted class on eval data')
plt.show()
```



```
[ ]: error_vec_2 = [0 if i[0]==i[1] else 1 for i in np.hstack((y_hat_2, y_eval))]  
plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==0 else 'r' for i in_  
    ↪error_vec_2])  
plt.title('errors')  
plt.show()  
  
print('Error: ' + str(sum(error_vec_2)))
```



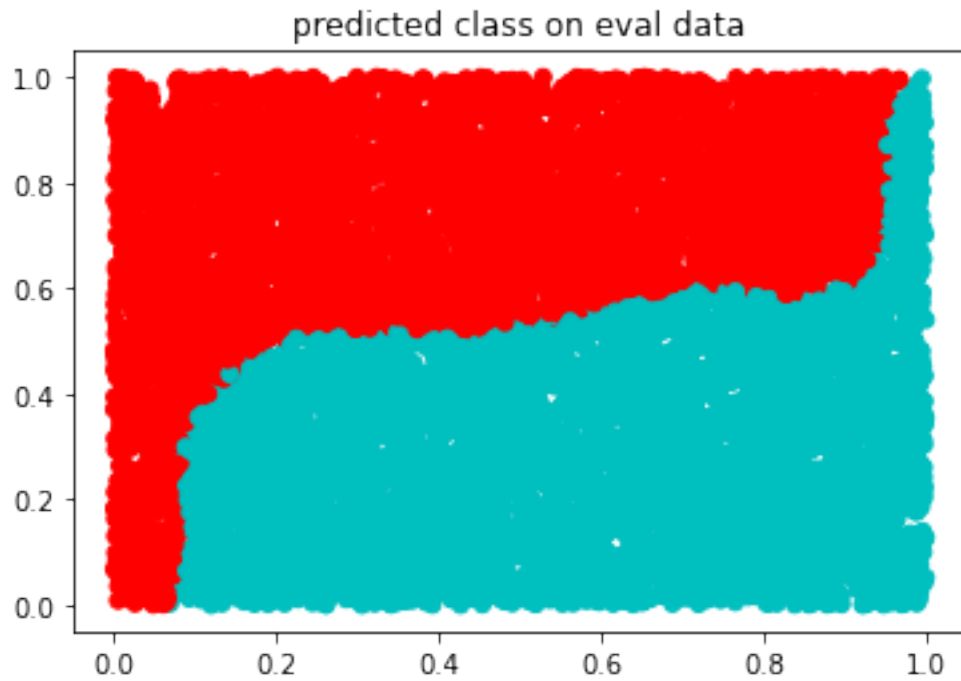
Error: 1066

## 6 3c)

```
[ ]: ## Classifier 3
x_train_3 = np.hstack((x_train**6,x_train**5,x_train**4,x_train**3,x_train**2,
    ↪x_train, np.ones((n_train,1)) ))
x_eval_3 = np.hstack((x_eval**6, x_eval**5, x_eval**4, x_eval**3, x_eval**2,
    ↪x_eval, np.ones((n_eval,1)) ))

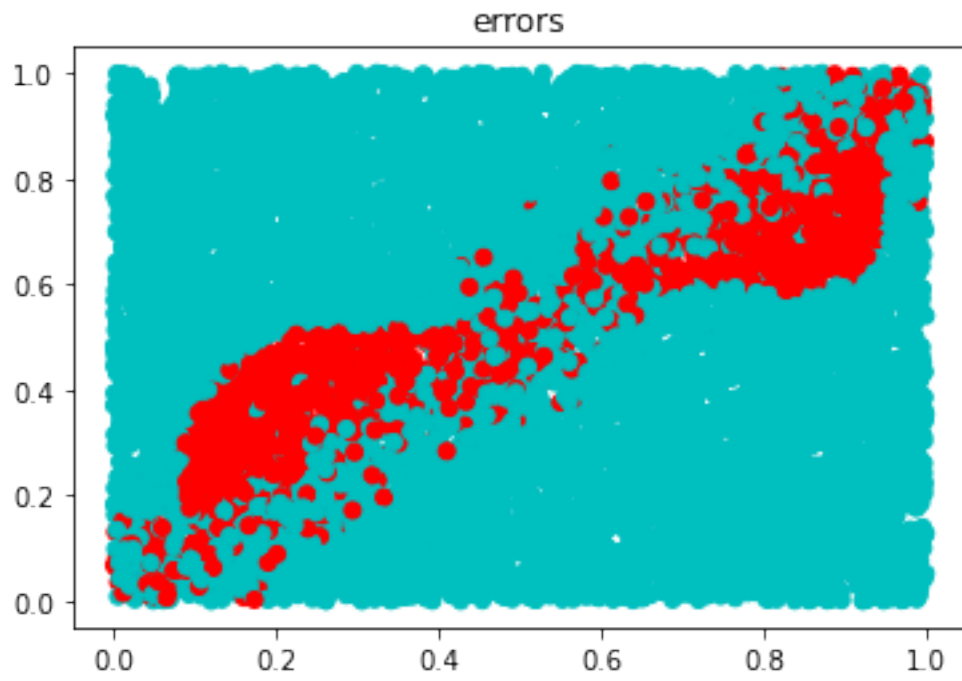
w_opt_3 = np.linalg.inv(x_train_3.transpose()@x_train_3)@x_train_3.
    ↪transpose()@y_train
y_hat_3 = np.sign(x_eval_3@w_opt_3)

plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==1 else 'r' for i in
    ↪y_hat_3[:,0]])
plt.title('predicted class on eval data')
plt.show()
```



```
[ ]: error_vec_3 = [0 if i[0]==i[1] else 1 for i in np.hstack((y_hat_3, y_eval))]
plt.scatter(x_eval[:,0],x_eval[:,1], color=['c' if i==0 else 'r' for i in_
    ↪error_vec_3])
plt.title('errors')
plt.show()

print('Error: ' + str(sum(error_vec_3)))
```



Error: 1677



## CS/ECE/ME532 Activity 8

*Estimated Time: 15 min for P1, 15 min for P2, 20 min for P3, 5 min for P4, 15 min for P5*

- 1. Binary linear classifiers.** Assume there are two possible labels,  $y = 1$  or  $y = -1$  associated with two features  $x_1$  and  $x_2$ . We consider several different linear classifiers  $\hat{y} = \text{sign}\{\mathbf{x}^T \mathbf{w}\}$  where  $\mathbf{x}$  is derived from  $x_1$  and  $x_2$  and  $\mathbf{w}$  are the classifier weights. Define the decision boundary of the classifier as the set  $\{x_1, x_2\}$  for which  $\mathbf{x}^T \mathbf{w} = 0$ . Let  $x_2$  be the vertical axis in your sketches and depict the interval  $0 \leq x_1, x_2 \leq 1$ .

$$\frac{1}{\frac{1}{4} + \frac{\sqrt{25}}{4}}$$

$$\frac{2}{\sqrt{21}} \begin{bmatrix} 1 \\ \frac{5}{2} \end{bmatrix}$$

- a) *Classifier 1.* Let  $\mathbf{x}^T = [x_1 \ x_2]$  and assume  $\mathbf{w} = \begin{bmatrix} 5 \\ -2 \end{bmatrix}$ .

$$5x_1 - 2x_2 = 0$$

$$x_2 = \frac{5}{2}x_1$$

- i. Sketch the decision boundary in the  $x_1$ - $x_2$  plane.
- ii. Does the decision boundary represent a subspace in  $\mathbb{R}^2$ ? Why or why not? If it represents a subspace, then find an orthonormal basis for the subspace.

*Yes because it goes through the origin*

- b) *Classifier 2.* Let  $\mathbf{x}^T = [x_1 \ x_2 \ 1]$  and assume  $\mathbf{w} = \begin{bmatrix} 5 \\ -2 \\ 1 \end{bmatrix}$ .

- i. Sketch the decision boundary in the  $x_1$ - $x_2$  plane.
- ii. Does the decision boundary represent a subspace in  $\mathbb{R}^2$ ? Why or why not? If it represents a subspace, then find an orthonormal basis for the subspace.

*Not a subspace because it doesn't contain the origin*

- c) *Classifier 3.* Let  $\mathbf{x}^T = [x_1^2 \ x_2 \ 1]$  and assume  $\mathbf{w} = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$ .

- i. Sketch the decision boundary in the  $x_1$ - $x_2$  plane.
- ii. Does the decision boundary represent a subspace in  $\mathbb{R}^2$ ? Why or why not? If it represents a subspace, then find an orthonormal basis for the subspace.

*No because it doesn't contain the origin*

- 2. Linear Classifier.** Download the script and the data file `classifier_data.mat`. This code trains linear classifiers using least squares. The scripts provided steps through the problems below.

Make sure to 'publish' your results and store them as a PDF file for submission.

- a) *Classifier 1.* Let  $\mathbf{x}^T = [x_1 \ x_2]$ . Briefly comment on the fit of the classifier to the decision boundary apparent in the evaluation data. Also identify the percent error based on the ratio of misclassified evaluation data points to the total number of evaluation data points.

*The classifier is linear but the decision boundary appears to be non-linear 11.02 11%.*

- b) *Classifier 2.* Consider squaring the original features, and also using them for classification, so that:  $\mathbf{x}^T = [x_1^2 \ x_2^2 \ x_1 \ x_2 \ 1]$ . This will allow for a curved decision boundary. Briefly comment on the fit of the classifier to the decision boundary apparent in the evaluation data. Also identify the percent error based on the ratio of misclassified evaluation data points to the total number of evaluation data points.

*The classifier is non-linear and the error rate*

- c) *Shortcoming of training using least squares as a loss function.* Training a classifier using the squared error as a loss function can fail when correctly labeled data points lie far from the decision boundary. A new dataset consisting of the first dataset, plus 1000 (correctly labeled) datapoints at  $x_1 = 0, x_2 = 3$  is created. What happens to the decision boundary when these new data points are included in training? What happens to the error rate if you move the 1000 data points to  $x_1 = 0, x_2 = 10$ ? Why does this happen?

*is 5%.*

*50 - 50%*

*shift of*

*the prob*

*rate*

*increases*

*to 20%*

3. *Overfitting.* Download the dataset `overfitting_data.mat`. You may find it helpful to adapt the code from the previous problem. The dataset has 50 data points for training, and 10,000 data points to be used for evaluation of the classifier. Each data point consists a two-dimensional feature vector  $\mathbf{x}$  and a label  $y \in \{-1, 1\}$ . The feature vector is a “noisy” version of the true underlying feature, which blurs the boundary between classes.

- a) Plot the training data using a scatter plot. Indicate the points  $y = -1$  using one color, and the points with  $y = 1$  with another.

- b) Plot the evaluation data using a scatter plot. Indicate the points labeled  $-1$  using one color, and the points labeled  $+1$  with another.

- c) Classifier 1. As before,  $\mathbf{x} = [x_1 \ x_2]^T$ , and  $y = \text{sign}(\mathbf{x}^T \mathbf{w})$ .

- Train the classifier using least squares to find the classifier weights  $\mathbf{w}$ . Apply the classifier to the evaluation data, and plot the data points using a scatter plot with different colors for different predicted labels.
- Plot the correctly predicted evaluation data points using one color, and incorrectly predicted points using a second color. How many errors are there?

*75%*

- d) Classifier 2. Let  $\mathbf{x} = [x_1^2 \ x_2^2 \ x_1 \ x_2 \ 1]^T$ , and  $y = \text{sign}(\mathbf{x}^T \mathbf{w})$ .

- Train the classifier using least squares to find the classifier weights  $\mathbf{w}$ . Apply the classifier to the evaluation data, and plot the data points using a scatter plot with different colors for different predicted labels.
- Plot the correctly predicted evaluation data points using one color, and incorrectly predicted points using a second color. How many errors are there?

*1066*

e) Classifier 3. Let  $\mathbf{x} = [x_1^6 \ x_2^6 \ x_1^5 \ x_2^5 \ \dots \ x_1 \ x_2 \ 1]^T$ , and  $y = \text{sign}(\mathbf{x}^T \mathbf{w})$ .

- i. Train the classifier using least squares to find the classifier weights  $\mathbf{w}$ . Apply the classifier to the evaluation data, and plot the data points using a scatter plot with different colors for different predicted labels.
- ii. Plot the correctly predicted evaluation data points using one color, and incorrectly predicted points using a second color. How many errors are there?

1677

f) Of the three classifiers, which one performs worse, and why?

the third because it is overfitted

4. A binary linear classifier based on three features  $x_1, x_2$ , and  $x_3$  is  $\hat{y} = \text{sign}\{\mathbf{x}^T \mathbf{w}\}$  where  $\mathbf{x}^T = [x_1 \ x_2 \ x_3]$ . Hence the decision boundary is the set  $\{x_1, x_2, x_3\}$  for which  $\mathbf{x}^T \mathbf{w} = 0$ .

The decision boundary for a two-dimensional classifier is a line. What type of geometric object is the decision boundary in three dimensions?

A plane

5. A decision boundary for a classification problem involving features  $x_1, x_2$ , and  $x_3$  is defined as  $\mathbf{x}^T \mathbf{w} = 0$  where  $\mathbf{x}^T = [x_1 \ x_2 \ x_3 \ 1]$ . Find  $\mathbf{w}$  so that the decision boundary is parallel to the  $x_1$ - $x_2$  plane and includes the point  $(x_1, x_2, x_3) = (0, 0, 1)$ .