# <u>ECE GY-6143 ML Homework 2 Solution</u>

**Yogya Sharma**
**(ys5250@nyu.edu)**

**1.**

Using Stochastic Gradient Descent (SGD) to calculate the model parameters for linear perceptron.

Algorithm for SGD:

$$
\begin{aligned}
& initialize\; t = 0 \;\; and\; \theta^0 = \vec{0} \\
& while\; not\; converged \{ \\
& \quad pick\; i \in \{1, \ldots, N\} \\
& \quad if \left( y_i x_i^T \theta^t \leq 0 \right) \quad \{ \;\; \theta^{t+1} = \theta^t + y_i x_i \\
& \qquad\qquad\qquad\qquad\quad\;\; t = t + 1 \qquad\qquad \} \}
\end{aligned}
$$

Formula for calculating Perceptron Loss (Risk):

$$R^{per}\left(\theta\right) = -\tfrac{1}{N} \sum\nolimits_{i \in misclassified} y_i \left(\theta^T x_i\right)$$

Formula used for calculating Binary Classification Error:

$$E(\theta) = \frac{N_{misclassified}}{N_{total}}$$

The loop in the code below is used to calculate the concurrent model parameters and corresponding Perceptron Loss and Binary Classification Error. The code exits the loop as soon as the binary classification error becomes zero, as in there are no misclassified labels calculated using the model parameters of the final iteration.

```
 import scipy.io
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#importing dataset
dataset = scipy.io.loadmat('data3.mat')

#creating a dataframe and adding the bias parameter x0 = 1 to it
df = pd.DataFrame(dataset['data'] , columns = ['x1' , 'x2' , 'y'])
```

```
df['x0'] = 1.0
df = df.iloc[ : , [3, 0, 1, 2]]

#Extractiong X and Y from the dataframe
X = pd.DataFrame(df.iloc[ : , :3])
Y = pd.DataFrame(df.iloc[ : , 3:4])

#initializing model parameter values using randomize operation
T = pd.DataFrame(np.random.randint(200 , size = (3)))

#declaring a list to store binary classifictaion error for each iteration
binaryClassificationError = list()

#declaring a list to store perceptron loss for each iteration
perceptronLoss = list()

misclassCheck = list()

#Setting up iteration counter
iterations = 0
check = 1
falseCheck = 1

#Starting to loop to calculate model parameters and corresponding error
and risk
while check:

  #Checking if there is any misclassifiction after the previous value of
model parameters
  #starts from the second iteration
  if iterations > 0:
    falseCheck = 0
    for g in range(len(misclassCheck[0])):
      if misclassCheck[0][g] == 'false':
        falseCheck = falseCheck+1

  if falseCheck != 0:

    count = 0
    iterations = iterations+1
    misclassCheck.clear()

    #calculating the predictions
    xT = pd.DataFrame(np.dot( X , T ) , columns = ['xT'])
```

```
    predictions = list()

    #appending the newly calculated labels in a list
    predictions.append(np.where(xT < 0, -1.0 , 1.0))

    pred = pd.DataFrame(predictions[0], columns = ['predictions'])

    #appending misclassCheck list by if the predicted value is true or
false
    misclassCheck.append(np.where(pred['predictions'] == Y['y'], 'true',
'false'))

    #calculating binary classifiction error
    for i in range(len(misclassCheck[0])):
      if misclassCheck[0][i] == 'false':
        count = count + 1

    error = count/len(X)

    binaryClassificationError.append(error)

    falseIndex = []

    #Checking which indexes have been misclassified
    falseIndex = np.where(misclassCheck[0] == 'false')[0]

    risk = 0
    err = 0

    #calculating perceptron loss
    for j in falseIndex:
      err = err + np.dot(xT.loc[[j]], Y.loc[[j]])[0][0]

    risk = -(err)/len(X)

    perceptronLoss.append(risk)

    #evaluating model parameters
    if count != 0:
      T = T + np.dot(X.iloc[[falseIndex[0]]].T , Y.iloc[[falseIndex[0]]] )
      check = 1
    else:
      check = 0
      print("Final model parameter values", T)
```

```
print("Number of iterations:", iterations)
#Plotting scatter plot for y values
figure, hlt = plt.subplots(1,3)
for r in [-1, 1]:
    row = np.where(df.y == r)
    hlt[0].scatter( df.iloc[row[0],1], df.iloc[row[0], 2], s= 0.5 )

hlt[0].set_title("Scatter Plot for Training Data")
hlt[0].set_xlabel("x1")
hlt[0].set_ylabel("x2")

#plotting decision boundary
decisionBoundary = - (float(T.iloc[1])/float(T.iloc[2])) * X.iloc[:,1] -
(float(T.iloc[0])/float(T.iloc[2]))
hlt[0].plot(X.iloc[:,1], decisionBoundary, 'y')

#plotting risk vs. iterations
hlt[1].plot(range(iterations), perceptronLoss, '.', color = 'red')
hlt[1].set_title("Risk vs. Number of Iterations")
hlt[1].set_xlabel("Number of Iterations")
hlt[1].set_ylabel("Risk")

#plotting binary classifictaion error vs. iterations
hlt[2].plot(range(iterations), binaryClassificationError, '.', color =
'blue')
hlt[2].set_title("Binary Classification Error vs. Number of Iterations")
hlt[2].set_xlabel("Number of Iterations")
hlt[2].set_ylabel("Binary Classification Error")

plt.subplots_adjust(left=0.3,
                    bottom=0.2,
                    right=3.0,
                    top=0.9,
                    wspace=0.4,
                    hspace=0.4)


plt.grid()
plt.show()
```

Flow of the code:

1. We set random model parameter values (T).
2. Calculate the predictions and store them in misclassCheck.
3. Check for what parameters the predictions are wrong and store them in falseIndex.
4. Choose the first index (from falseIndex) where the prediction was incorrect.

5. Calculate the perceptron loss and binary classification error.
6. Correct our model parameter values (T) according to those input values.
7. Repeat step 2 to 6, until binary classification error becomes 0, as in till there are no wrong predictions.

Number of iterations = 3286

Values of model parameters:

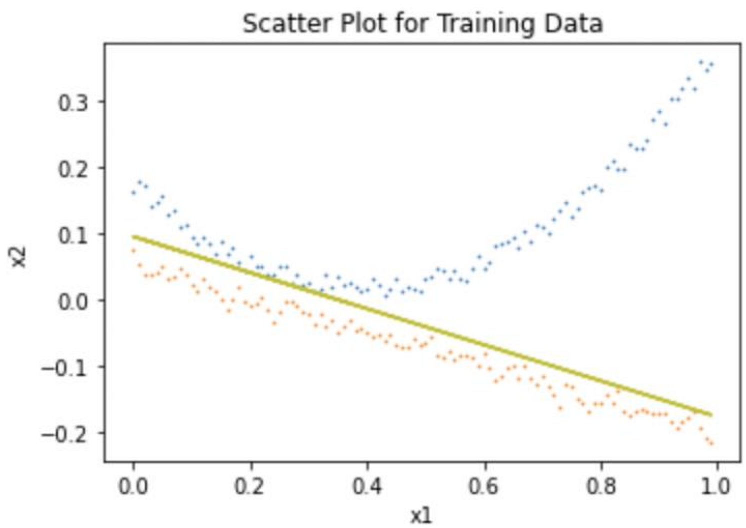
$$x0 = 1.000000$$
$$x1 = -2.870000$$
$$x2 = -10.601136$$

Decision boundary is given as,

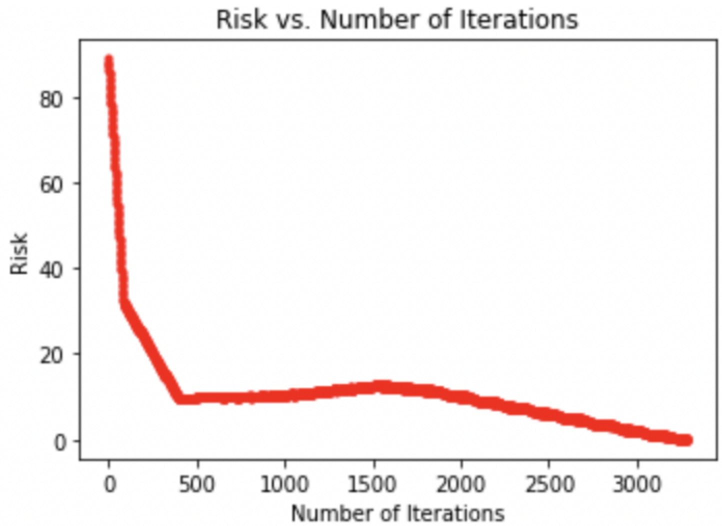
$$\Theta^T x = 0$$

$$1.000000 - 2.870000 * x1 - 10.601136 * x2 = 0$$

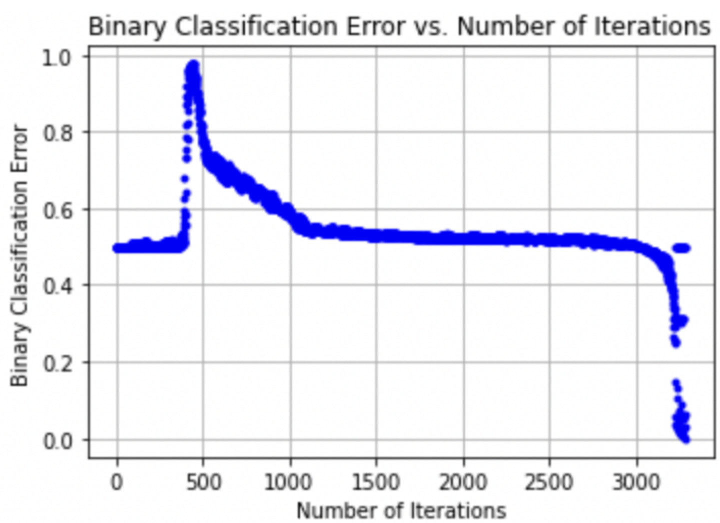
$$\mathbf{x2 = -0.27056\ x1 + 0.09433}$$




Here, we can see that the obtained decision boundary separates the data with 100% accuracy.



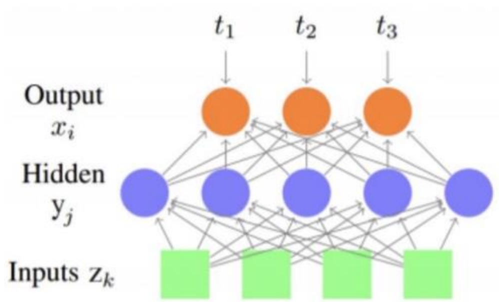

As shown in the above figure we can see that the risk reduces with each iteration till it reaches 0. The data took 3286 iterations to converge, with final perceptron loss as 0.




As the data converges, we can see that that on the last iteration of the code, the binary classification error is 0.

**2.**

Answer 2)




Assumption: Single training example

a) $$E = -\sum_i \left(t_i \log(x_i) + (1-t_i)\log(1-x_i)\right)$$

t is the target, and logistic activation function for output units is

① → $$x_i = \frac{1}{1+e^{-s_i}}, \text{ where } s_i = \sum_j y_j w_{ji}$$

where $w_{ji}$ denotes the weight of the edge between $j^{th}$ hidden unit and $i^{th}$ output unit.

Assuming that all layers use logistic activation $f^n$

② $$y_j = \frac{1}{1+e^{-s_j}}, \text{ where } s_j = \sum_k z_k w_{kj}$$

For outer derivative,

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial x_i} \frac{\partial x_i}{\partial w_{ji}} \rightarrow \text{(A)}$$

$$\frac{\partial E}{\partial x_i} = - \frac{t_i}{x_i} + (1-t_i)\frac{1}{1-x_i} \times(-1)$$

$$= - \frac{t_i}{x_i} - \frac{(1-t_i)}{1-x_i} = \left[\frac{t_i - \cancel{t_i x_i} - x_i + \cancel{t_i x_i}}{x_i(1-x_i)}\right]$$

$$(4) \rightarrow \quad = - \frac{(t_i - x_i)}{x_i(1-x_i)}$$

$$\frac{\partial x_i}{\partial w_{ji}} = \frac{\partial x_i}{\partial s_i} \frac{\partial s_i}{\partial w_{ji}} \rightarrow (5)$$

$$\frac{\partial x_i}{\partial s_i} = \frac{e^{-s_i}}{(1+e^{-s_i})^2} \; ; \; \frac{\partial s_i}{\partial w_{ji}} = y_j \rightarrow (7)$$

from (1) $\rightarrow = x_i(1-x_i) \rightarrow (6)$

Evaluating (5) using (6) and (7)

$$\frac{\partial x_i}{\partial s_i} = x_i(1-x_i) \cdot y_j \rightarrow (8)$$

Evaluating (A) using (4) and (8)

$$\frac{\partial E}{\partial w_{ji}} = \frac{(x_i - t_i)}{\cancel{x_i(1-x_i)}} \times \cancel{x_i(1-x_i)} \cdot y_j = \underbrace{(x_i - t_i)}_{\delta_i} y_j$$

$$\boxed{\frac{\partial E}{\partial w_{ji}} = \delta_i y_j}$$

For inner derivative,

$$\frac{\partial E}{\partial w_{kj}} = \left(\sum_i \frac{\partial E}{\partial s_j}\right) \frac{\partial s_j}{\partial w_{kj}}$$

$$= \sum_i \frac{\partial E}{\partial s_i} \frac{\partial s_i}{\partial s_j} \frac{\partial s_j}{\partial w_{kj}} = \frac{\partial E}{\partial s_i} \frac{\partial s_i}{\partial y_j} \frac{\partial y_j}{\partial s_j} \frac{\partial s_j}{\partial w_{kj}}$$

$$= \sum_i \frac{\partial E}{\partial x_i} \frac{\partial x_i}{\partial s_i} \frac{\partial s_i}{\partial y_j} \frac{\partial y_j}{\partial s_j} \frac{\partial s_j}{\partial w_{kj}}$$

From (4) $\rightarrow$ $\frac{\partial E}{\partial x_i} = \frac{x_i - t_i}{x_i(1-x_i)}$

From (6) $\rightarrow$ $\frac{\partial x_i}{\partial s_i} = x_i(1-x_i)$

$$\frac{\partial s_i}{\partial y_j} = \frac{\partial \sum_j w_{ji} y_j}{\partial y_j} = w_{ji} \rightarrow (9)$$

From (6) we can similarly calculate $\partial y_j / \partial s_j$

$$\frac{\partial y_j}{\partial s_j} = y_j(1-y_j) \rightarrow (10)$$

$$\frac{\partial s_j}{\partial w_{kj}} = \frac{\partial \sum w_{kj} z_k}{\partial w_{kj}} = z_k \rightarrow (11)$$

from ④, ⑥, ⑨, ⑩ and ⑪

$$\frac{\partial E}{\partial w_{kj}} = \sum_i \frac{x_i - t_i}{\cancel{x_i(1-x_i)}} \times \cancel{x_i(1-x_i)} \times w_{ji} \times y_j(1-y_j) \times z_k$$

$$= (x_i - t_i)\, y_j(1-y_j)\, w_{ji} \times z_k$$

$$\boxed{\frac{\partial E}{\partial w_{kj}} = \underbrace{\sum_i (x_i - t_i)\, y_j(1-y_j)\, w_{ji}}_{\delta_j}\, z_k}$$

b) $E = -\sum_i t_i \log(x_i)$

activation function, $x_i = \frac{e^{s_i}}{\sum_{c=1}^{m} e^{s_c}}$ where $s_i = \sum_j y_j w_{ji}$

For outer derivative,

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial x_i} \cdot \frac{\partial x_i}{\partial w_{ji}} \longrightarrow \text{A}$$

$$\frac{\partial E}{\partial x_i} = \frac{\partial\left(-\sum_i t_i \log(x_i)\right)}{\partial x_i} = -t_i/x_i \rightarrow 1$$

$$\frac{\partial x_i}{\partial w_{ji}} = \frac{\partial x_i}{\partial s_i} \cdot \frac{\partial s_i}{\partial w_{ji}}$$

$$\frac{\partial x_i}{\partial s_i} = \frac{\partial}{\partial s_i}\left( e^{s_i} / \sum_i e^{s_i} \right)$$

$$= -\frac{\left[ e^{s_i} \left( \sum_i e^{s_i} \right) \right]}{\left( \sum_i e^{s_i} \right)^2} + \frac{e^{s_i} \left( \sum_i e^{s_i} \right)}{\left( \sum e^{s_i} \right)^2}$$

$$= \frac{e^{s_i}}{\left( \sum_i e^{s_i} \right)} - \frac{{e^{s_i}}^2}{\left( \sum e^{s_i} \right)^2}$$

$$= x_i - x_i^2 = x_i (1 - x_i) \longrightarrow 2$$

$$\frac{\partial s_i}{\partial w_{ji}} = y_j \rightarrow 3$$

calculating A using 1, 2 and 3

$$\frac{\partial E}{\partial w_{ji}} = -\frac{t_i}{\cancel{x_i}} \cancel{x_i}(1 - x_i) y_j$$

$$= - t_i (1 - x_i) y_j = \delta_i y_j$$

$$\boxed{\frac{\partial E}{\partial w_{ji}} = - t_i (1 - x_i) y_j = \delta_i y_j}$$

For the inner derivative,

$$\frac{\partial E}{\partial w_{kj}} = \sum_i \frac{\partial E}{\partial s_j} \cdot \frac{\partial s_j}{\partial w_{kj}} \longrightarrow (B)$$

$$= \sum_i \frac{\partial E}{\partial s_i} \frac{\partial s_i}{\partial s_j} \frac{\partial s_j}{\partial w_{kj}}$$

As derived before, $\frac{\partial E}{\partial s_i} = \delta_i$

$$\frac{\partial E}{\partial w_{kj}} = \delta_i \frac{\partial s_i}{\partial s_j} \frac{\partial s_j}{\partial w_{kj}} \longrightarrow (4)$$

$$s_i = \sum_j y_j w_{ji} \quad \& \quad y_j = \frac{1}{1+e^{-s_j}}$$

In (4), $\frac{\partial s_i}{\partial s_j} = \frac{\partial s_i}{\partial y_j} * \frac{\partial y_j}{\partial s_j}$

$$\frac{\partial s_i}{\partial y_j} = w_{ji} \quad \text{and} \quad \frac{\partial y_j}{\partial s_j} = y_j (1-y_j)$$

$$\frac{\partial s_i}{\partial s_j} = w_{ji}\, y_j (1-y_j) \rightarrow (5)$$

In (4), $\frac{\partial s_j}{\partial w_{kj}} = z_k \longrightarrow (6)$

Calculating (B), using (4), (5) and (6)

$$\boxed{\frac{\partial E}{\partial \omega_{kj}} = \sum_i \underbrace{\delta_i \, w_{ji} \, y_j \, (1-y_j)}_{\delta_j} z_k}$$

Answer 3 on next page

Answer 3) Discrete Distribution : $\{ P_K \mid K = 1, 2 \dots N \}$

Entropy : $$\underbrace{H = -\sum_{K=1}^{N} P_K \log P_K}$$

we need to maximise this

Using lagranges Theorem,

$$L(P, \lambda) = -\sum_{K=1}^{N} P_K \log P_K + \lambda \left( \sum_{K=1}^{N} P_K - 1 \right)$$

Taking partial derivatives and equating them to 0.

$$\frac{\partial L(P, \lambda)}{\partial P} = 0$$

$$-\log P_K - 1 + \lambda = 0 \Rightarrow P_K = e^{\lambda - 1}$$

$$P_1 = P_2 = \dots \dots P_N = 1/N \rightarrow ①$$

Thus all $P_K$ are equal.

from ①, we see that $\sum_{K=1} P_K = 1$, as $N \times \frac{1}{N} = 1$.

$$P_K = 1/N$$

The maximum entropy is

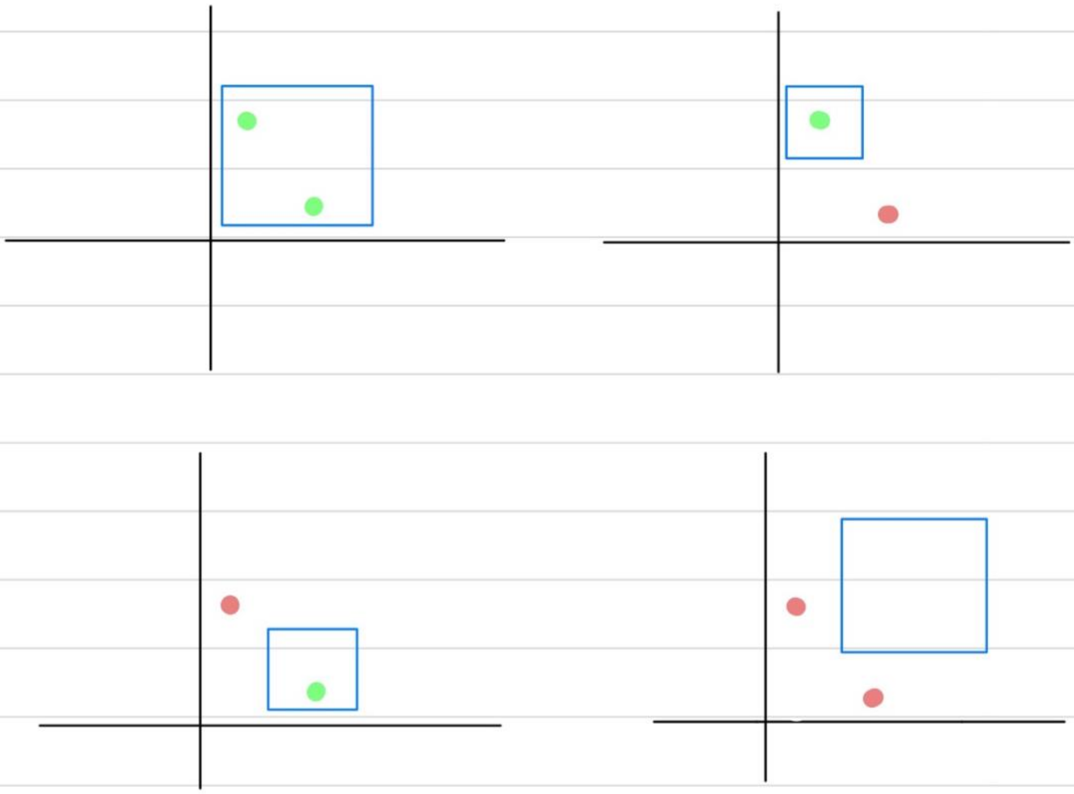
$$H_{max}(P) = -\sum_{K=1}^{N} \frac{1}{N} \log \frac{1}{N} = \log N$$

Thus $P_K = 1/N$ maximizes the entropy.

Distribution is $P_1 = P_2 = P_3 = \dots = P_N = 1/N$

Answer 4)

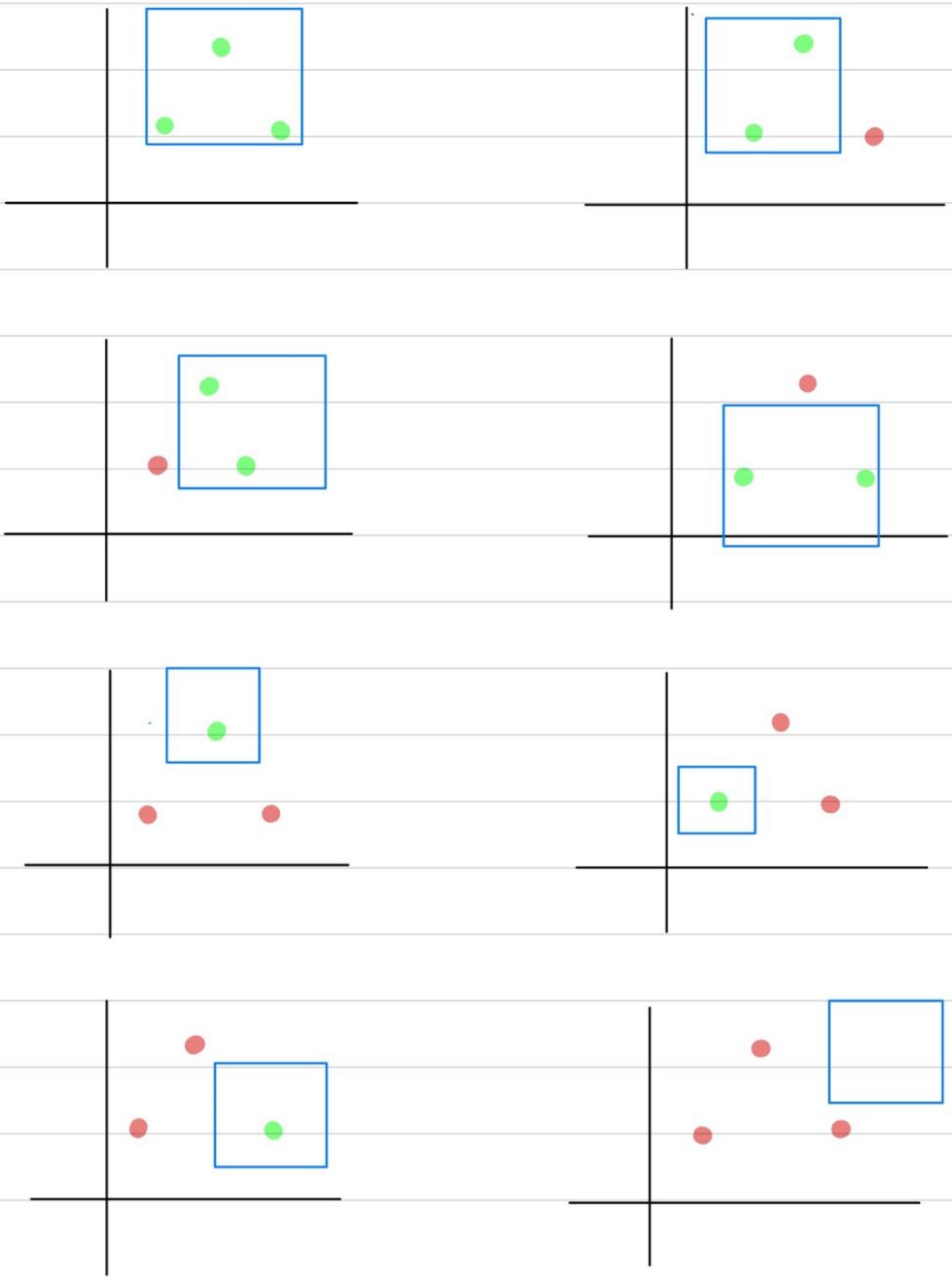
For VC dimension = 2



For VC dimension = 0, axis aligned squares can shatter all formations.

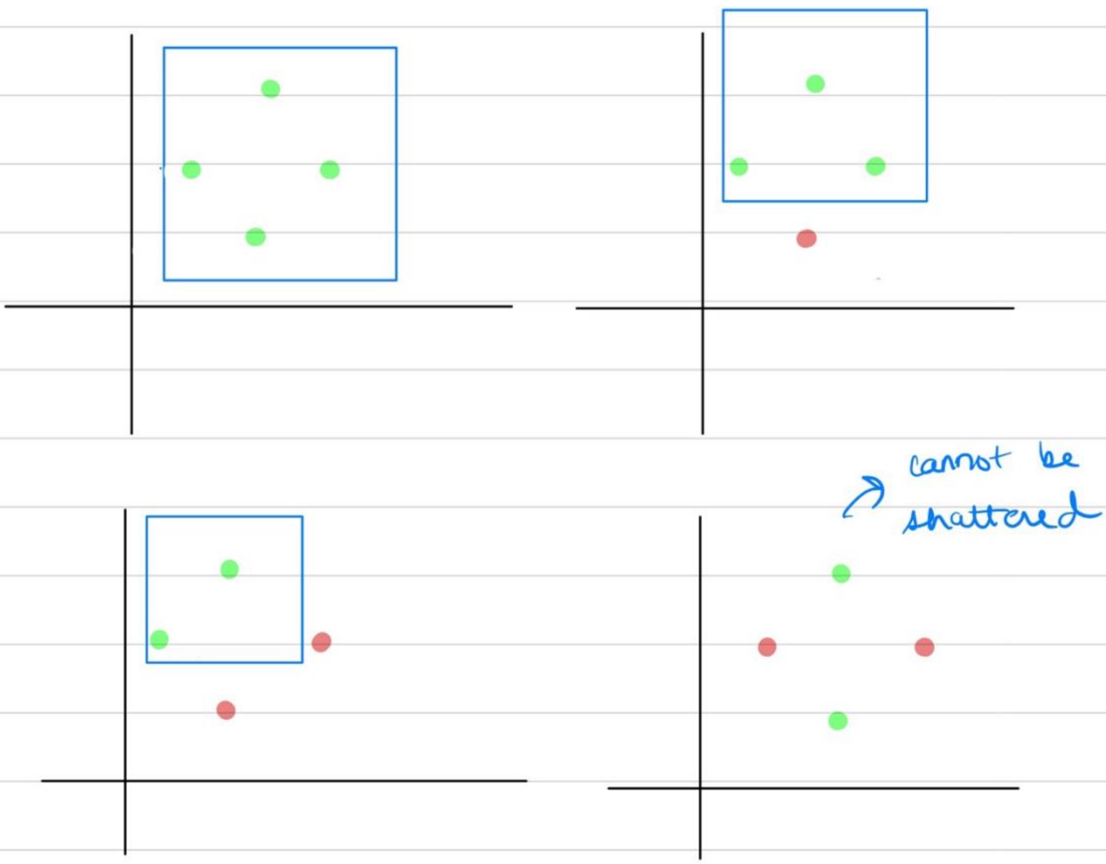
lets check for VC dimension = 3

For VC dimension = 3



We can see that 3 data points can be shatterd for all possible labels.

Lets check for VC dimension = 4




So, the VC dimension for axis aligned squares is 3.