

**NATIONAL UNIVERSITY OF SCIENCES & TECHNOLOGY**

**Machine Learning (CS-471)**

Quiz 1

*Estimating Strict and General Hypothesis*

**Submission Details**

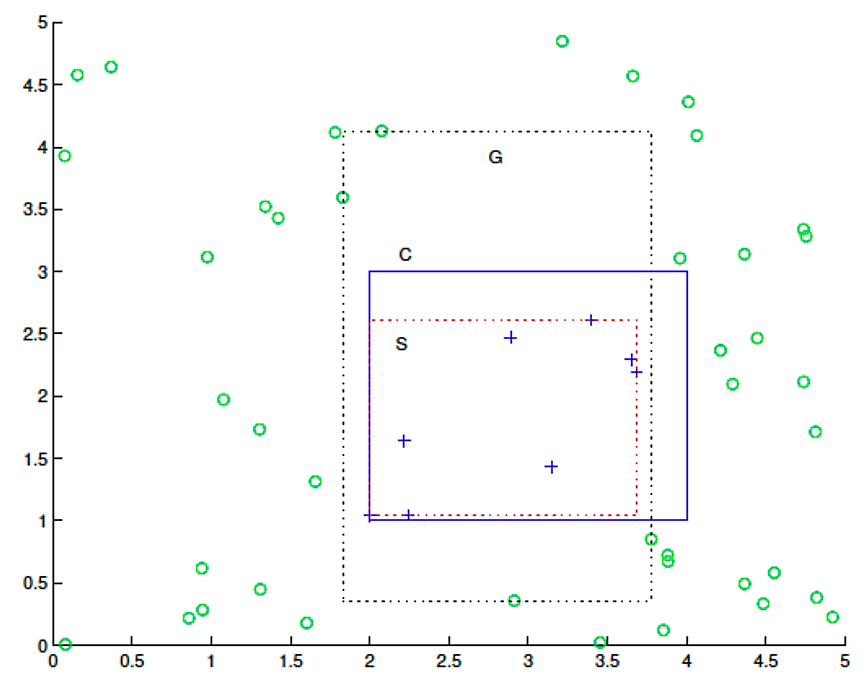
|  |  |
| --- | --- |
| Name | CMS ID |
| Muhammad Umer | 345834 |
| Submitted to: | Dr. Ahmad Salman |
| Class: | BEE-12 |
| Semester: | 7th |
| Dated: | 27/9/2023 |

# **Task**

## Explanation

As explained in the lecture notes, for the depicted training data in the figure, estimate the hypotheses S and G, as they are shown. The process of calculating S and G should be automatic and not hard-coded. This means if we have to change the training data points, the algorithm should automatically find the S and G for that data too. You can approximate the data coordinates from the given figure. Once done, create a test data with similar ranges but randomly generated coordinate for both positive and negative examples. The new test data should be mapped on the S and G hypotheses of training data. So, on test data, the code should automatically find out:

1. Percentage of positive points falling in S
2. Percentage of negative points falling in S
3. Percentage of positive points falling in G
4. Percentage of negative points falling in G



## Deliverables

Python source code, results, and relevant scatter plots to show hypotheses on training data and test data. Make a single PDF file and upload to LMS

# Solution

## Generation of Data Samples

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.patches import Rectangle

*# Small tau value for adding spacing*

tau = 0.15

*# Generate positive samples*

positive\_samples = np.random.uniform(

*low*=(2 + tau, 1 + tau), *high*=(4 - tau, 3 - tau), *size*=(20, 2)

)

*# Generate negative samples*

negative\_samples = np.random.uniform(*low*=[0, 0], *high*=[5, 5], *size*=(80, 2))

*# Remove negative samples that are inside the positive region*

negative\_samples = negative\_samples[

    ~(

        (negative\_samples[:, 0] > 2 - tau \* 2)

        & (negative\_samples[:, 0] < 4 + tau \* 2)

        & (negative\_samples[:, 1] > 1 - tau \* 2)

        & (negative\_samples[:, 1] < 3 + tau \* 2)

    )

]

## Finding Strict (S) and General (G) Hypothesis

*# Find strict class boundary region from positive samples*

strict\_min\_x = np.min(positive\_samples[:, 0])

strict\_max\_x = np.max(positive\_samples[:, 0])

strict\_min\_y = np.min(positive\_samples[:, 1])

strict\_max\_y = np.max(positive\_samples[:, 1])

*# Find most general class boundary region from negative samples*

*# General class boundary is the extension of strict class boundary lines of each of the 4 sides until it hits a negative sample*

general\_min\_x = []

general\_max\_x = []

general\_min\_y = []

general\_max\_y = []

for sample in negative\_samples:

    if (

        sample[0] > strict\_min\_x

        and sample[0] < strict\_max\_x

        and sample[1] > strict\_max\_y

    ):

        general\_max\_y.append(sample[1])

for sample in negative\_samples:

    if (

        sample[0] > strict\_min\_x

        and sample[0] < strict\_max\_x

        and sample[1] < strict\_min\_y

    ):

        general\_min\_y.append(sample[1])

if not general\_max\_y == []:

    general\_max\_y = np.min(general\_max\_y)

else:

    general\_max\_y = strict\_max\_y

if not general\_min\_y == []:

    general\_min\_y = np.max(general\_min\_y)

else:

    general\_min\_y = strict\_min\_y

for sample in negative\_samples:

    if (

        sample[1] > general\_min\_y

        and sample[1] < general\_max\_y

        and sample[0] > strict\_max\_x

    ):

        general\_max\_x.append(sample[0])

for sample in negative\_samples:

    if (

        sample[1] > general\_min\_y

        and sample[1] < general\_max\_y

        and sample[0] < strict\_min\_x

    ):

        general\_min\_x.append(sample[0])

if not general\_max\_x == []:

    general\_max\_x = np.min(general\_max\_x)

else:

    general\_max\_x = strict\_max\_x

if not general\_min\_x == []:

    general\_min\_x = np.max(general\_min\_x)

else:

    general\_min\_x = strict\_min\_x

## Scatter Plots (Training Data ~ Estimation of {S, G} Hypothesis)

*# Plot positive and negative samples*

fig, ax = plt.subplots()

ax.scatter(

    positive\_samples[:, 0], positive\_samples[:, 1], *label*="Positive Samples", *marker*="+"

)

ax.scatter(

    negative\_samples[:, 0], negative\_samples[:, 1],

*label*="Negative Samples",

*facecolors*="none",

*edgecolors*="g",

)

*# Plot the strict class boundary region*

S = [strict\_min\_x, strict\_min\_y, strict\_max\_x, strict\_max\_y]

ax.add\_patch(

    Rectangle(

        (strict\_min\_x, strict\_min\_y), strict\_max\_x - strict\_min\_x,

        strict\_max\_y - strict\_min\_y, *fill*=False, *lw*=0.75, *linestyle*="--",

*edgecolor*="r",

    )

)

ax.text(strict\_min\_x + 0.05, strict\_max\_y - 0.25, "S", *fontsize*=12, *color*="r")

*# Plot the general class boundary region*

G = [general\_min\_x, general\_min\_y, general\_max\_x, general\_max\_y]

ax.add\_patch(

    Rectangle(

        (general\_min\_x, general\_min\_y), general\_max\_x - general\_min\_x,

        general\_max\_y - general\_min\_y, *fill*=False, *lw*=0.75, *linestyle*="--",

*edgecolor*="black",

    )

)

ax.text(general\_min\_x + 0.05, general\_max\_y - 0.25, "G", *fontsize*=12, *color*="black")

*# Plot the class boundary*

ax.add\_patch(Rectangle((2, 1), 2, 2, *fill*=False, *edgecolor*="b", *lw*=1))

ax.text(2.05, 2.75, "C", *fontsize*=12, *color*="b")

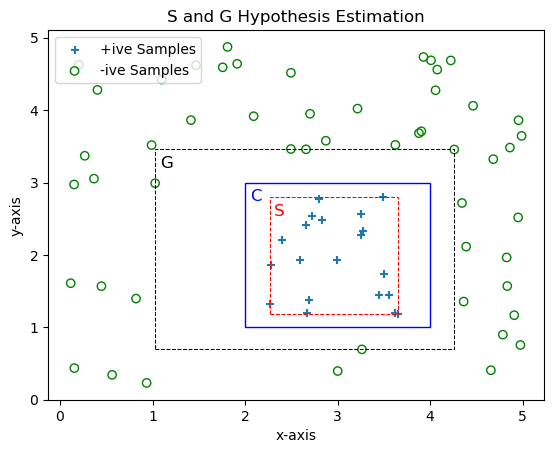
ax.set\_xlabel("x-axis")

ax.set\_ylabel("y-axis")

ax.legend(*loc*="upper left")

ax.set\_title("S and G Hypothesis Estimation")

plt.show()



## Mapping Hypothesis to Test Data

*# Small tau value for adding spacing*

tau = 0.15

*# Generate positive samples*

test\_positive\_samples = np.random.uniform(

*low*=(2 + tau, 1 + tau), *high*=(4 - tau, 3 - tau), *size*=(15, 2)

)

*# Generate negative samples*

test\_negative\_samples = np.random.uniform(*low*=[0, 0], *high*=[5, 5], *size*=(60, 2))

*# Remove negative samples that are inside the positive region*

test\_negative\_samples = test\_negative\_samples[

    ~(

        (test\_negative\_samples[:, 0] > 2 - tau \* 2)

        & (test\_negative\_samples[:, 0] < 4 + tau \* 2)

        & (test\_negative\_samples[:, 1] > 1 - tau \* 2)

        & (test\_negative\_samples[:, 1] < 3 + tau \* 2)

    )

]

*# Compute the percentages*

positive\_points\_in\_S = 0

negative\_points\_in\_S = 0

positive\_points\_in\_G = 0

negative\_points\_in\_G = 0

for sample in test\_positive\_samples:

    if (

        sample[0] >= strict\_min\_x and sample[0] <= strict\_max\_x

        and sample[1] >= strict\_min\_y and sample[1] <= strict\_max\_y

    ):

        positive\_points\_in\_S += 1

    if (

        sample[0] >= general\_min\_x and sample[0] <= general\_max\_x

        and sample[1] >= general\_min\_y and sample[1] <= general\_max\_y

    ):

        positive\_points\_in\_G += 1

for sample in test\_negative\_samples:

    if (

        sample[0] >= strict\_min\_x and sample[0] <= strict\_max\_x

        and sample[1] >= strict\_min\_y and sample[1] <= strict\_max\_y

    ):

        negative\_points\_in\_S += 1

    if (

        sample[0] >= general\_min\_x and sample[0] <= general\_max\_x

        and sample[1] >= general\_min\_y and sample[1] <= general\_max\_y

    ):

        negative\_points\_in\_G += 1

print(

    "Percentage of positive points falling in S: ",

    positive\_points\_in\_S / len(test\_positive\_samples),

)

print(

    "Percentage of negative points falling in S: ",

    negative\_points\_in\_S / len(test\_negative\_samples),

)

print(

    "Percentage of positive points falling in G: ",

    positive\_points\_in\_G / len(test\_positive\_samples),

)

print(

    "Percentage of negative points falling in G: ",

    negative\_points\_in\_G / len(test\_negative\_samples),

)

*# Plot scatter plot of test samples*

fig, ax = plt.subplots()

ax.scatter(

    test\_positive\_samples[:, 0], test\_positive\_samples[:, 1],

*label*="Test +ive Samples", *marker*="+",

)

ax.scatter(

    test\_negative\_samples[:, 0], test\_negative\_samples[:, 1],

*label*="Test -ive Samples", *facecolors*="none", *edgecolors*="g",

)

*# Plot the strict class boundary region*

ax.add\_patch(

    Rectangle(

        (strict\_min\_x, strict\_min\_y), strict\_max\_x - strict\_min\_x,

        strict\_max\_y - strict\_min\_y, *fill*=False, *lw*=0.75,

*linestyle*="--", *edgecolor*="r",

    )

)

ax.text(strict\_min\_x + 0.05, strict\_max\_y - 0.25, "S", *fontsize*=12, *color*="r")

*# Plot the general class boundary region*

ax.add\_patch(

    Rectangle(

        (general\_min\_x, general\_min\_y), general\_max\_x - general\_min\_x,

        general\_max\_y - general\_min\_y, *fill*=False, *lw*=0.75,

*linestyle*="--", *edgecolor*="black",

    )

)

ax.text(general\_min\_x + 0.05, general\_max\_y - 0.25, "G", *fontsize*=12, *color*="black")

*# Plot the class boundary*

ax.add\_patch(Rectangle((2, 1), 2, 2, *fill*=False, *edgecolor*="b", *lw*=1))

ax.text(2.05, 2.75, "C", *fontsize*=12, *color*="b")

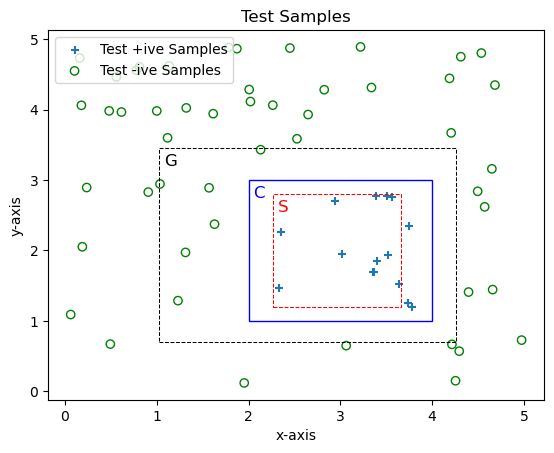
ax.set\_xlabel("x-axis")

ax.set\_ylabel("y-axis")

ax.legend(*loc*="upper left")

ax.set\_title("Test Samples")

plt.show()



**Percentage Results**

Percentage of positive points falling in S:  80.0

Percentage of negative points falling in S:  0.0

Percentage of positive points falling in G:  100.0

Percentage of negative points falling in G:  12.0