חישוביות וקוגניציה תר' 2   יובל פרידמן 204169320

(1.1)

Step:

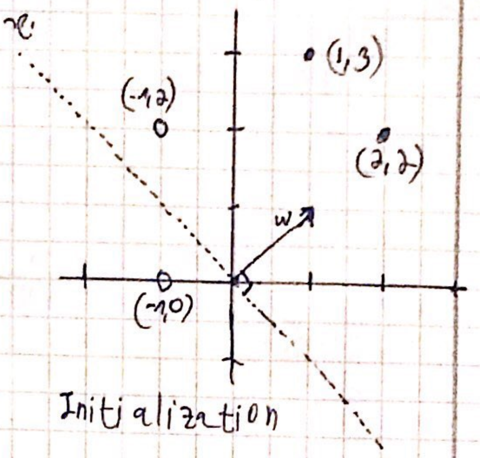
- $A: wx^1 = 2+2 = sgn(y^1)$
- $B: wx^2 = 1+3 = sgn(y^2)$
- $C: wx^3 = -1+0 = sgn(y^3)$
- $D: wx^4 = -1+2 \neq sgn(y^4) \Rightarrow$

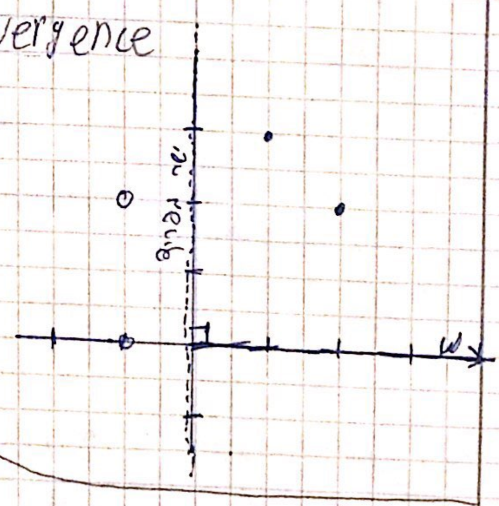
$$w = w + y^4 x^4 = \begin{bmatrix}1\\1\end{bmatrix} + (-1)\begin{bmatrix}-1\\2\end{bmatrix} = \begin{bmatrix}2\\-1\end{bmatrix}$$

- $E: wx^1 = 4-2 = sgn(y^1)$
- $F: wx^2 = 2-3 \neq sgn(y^2) \Rightarrow$

$$w = w + y^2 x^2 = \begin{bmatrix}2\\-1\end{bmatrix} + 1\begin{bmatrix}1\\3\end{bmatrix} = \begin{bmatrix}3\\2\end{bmatrix}$$

- $G: wx^3 = -3+0 = sgn(y^3)$
- $H: wx^4 = -3+4 \neq sgn(y^4) \Rightarrow w = \begin{bmatrix}3\\2\end{bmatrix} + (-1)\begin{bmatrix}-1\\2\end{bmatrix} = \begin{bmatrix}4\\0\end{bmatrix}$
- $I: wx^1 = 8+0 = sgn(y^1)$
- $J: wx^2 = 4+0 = sgn(y^2)$
- $K: wx^3 = -4+0 = sgn(y^3)$
- $L: wx^4 = -4+0 = sgn(y^4)$





(1.2.1)

$y=1$ אמ"מ $2x_1+x_2>0$

$\Longleftrightarrow$

$y=1$ אמ"מ $x_2>-2x_1$

כיוון שהמפריד הוא מקבע הקיצון $x_2=-2x_1$

ולכן w האפשרי הוא כל וקטור שמאונך לישר זה, כלומר שהשיפוע הוא $m=\frac{1}{2}$, למשל $w=\begin{bmatrix}2\\1\end{bmatrix}$.

b יהיה שווה ל-0 כיון שהישר המפריד עובר בראשית.

ניתן לראות גם שאם נבחר כך את w,b נקבל שהפרספטרון מוגדר:

$$\hat{y} = sgn(w_1x_1+w_2x_2+b) = sgn(2x_1+x_2) = y$$

שווה לסוג של הפרספטרון

(1.2.2)

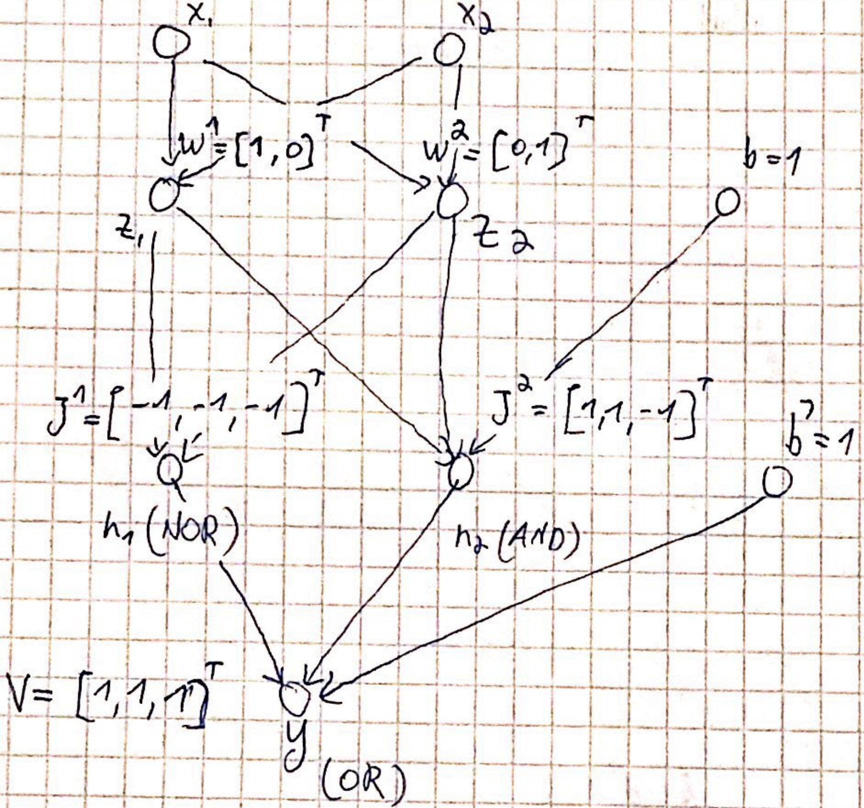
$$(-x_1+3x_2-4>0 \text{ אמ"מ } y=1) \Longleftrightarrow (x_1<3x_2-4 \text{ אמ"מ } y=-1)$$

באופן דומה לסעיף הקודם, נבחר כך לכן הפרספטרון: $w=\begin{bmatrix}-1\\3\end{bmatrix}, b=4$

$$\hat{y} = sgn(w_1x_1+w_2x_2+b) = sgn(-x_1+3x_2+4) = y$$

(1.3)

נממש את הפונ' על ידי שתי שכבות ביניים:
הראשונה משנה רק את סימן הקלט (ללא גודל) והשנייה תממש
את הפונקציות הלוגיות AND ו- NOR.
הפונקציה NXOR היא פונקציית OR מורכבת של
שתי הפונקציות AND ו- NOR, לכן שכבת הפלט תממש את OR.



השכבה הראשונה (Z) משנה את סימן.

השכבה השנייה: h_1 חיובי רק כאשר z_1, z_2 שליליים. $sgn(J^{1T}[z,b]) = h_1$

$h_2 = sgn(J^{2T}[z,b])$ חיובי רק כאשר z_1, z_2 חיוביים

לא הגיוני ש- h_1, h_2 יהיו שניהם חיוביים, אך עבור לפחות
1 בהגדרה ומכך מכאן יהיה חיובי, לכן זה מתיר לנו לממש
"OR": $y = sgn(V^T[h,b])$.
רק אם h_1, h_2 יהיו שליליים, משמעות בהגדרה היא שבאופסים
הפוכי סימן הפלט 1-.

ex2_computation_and_cognition

November 18, 2019

[0]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
import math
```

Q1 Implementation of perceptron learning algorithm

[0]:
```
# points_mat: nXp matrix of floats
# label: vecotr belongs to {-1,+1}^n
# w_init: number which will be the entries of the initial vector
# report: True if you want to print details of the algo's steps
def perceptron(points_mat,label, w_init=0, report=False):
  w = np.repeat(float(w_init),points_mat.shape[0])
  is_changed = True
  step=0
  while is_changed:
    is_changed = False
    for idx, point in enumerate(points_mat.T):
      step+=1
      if report:
        print("step",step)
        print("w=",w, "x=",point, "prod=",w @ point)
      if np.sign(w @ point) * label[idx] <=0:
        if report:
          print("!= sign", label[idx], "w=",label[idx]*point)
        w += label[idx]*point
        is_changed = True
      elif report:
        print("== sign", label[idx])
  return w
```

Q2

[0]:
```
# Function for sampling a 2XP matrix and a 1XP vector label
# The label rule is 1 if x1>x2, -1 O.W
def sample_two_dim_data(P):
```

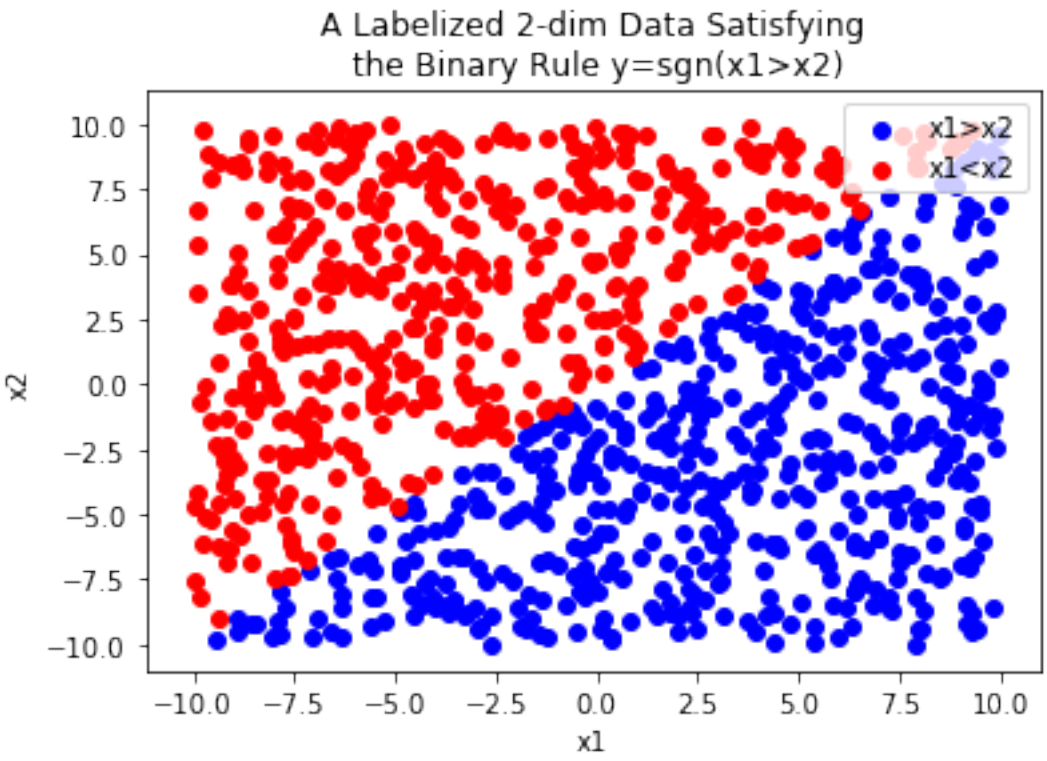
```
  points = np.random.uniform(low=-10,high=10, size=(2,P))
  label = np.apply_along_axis(lambda point: np.sign(point[0] - point[1]),
                              axis=0, arr=points)
  return points, label
```

Sample points and label

[0]:
```
mat, la = sample_two_dim_data(P=1000)
data = np.row_stack((mat, la))
blues = data[:,data[2,:] == 1]
reds = data[:,data[2,:] == -1]
```

Plot the points and colour their label

[0]:
```
fig, ax = plt.subplots()
ax.scatter(blues[0,:],blues[1,:], c='blue', label = 'x1>x2')
ax.scatter(reds[0,:],reds[1,:], c='red', label = 'x1<x2')
ax.title.set_text("A Labelized 2-dim Data Satisfying\n the Binary Rule␣
 ↪y=sgn(x1>x2)")
ax.set_xlabel("x1")
ax.set_ylabel("x2")
plt.legend()
plt.show()
```



Q3

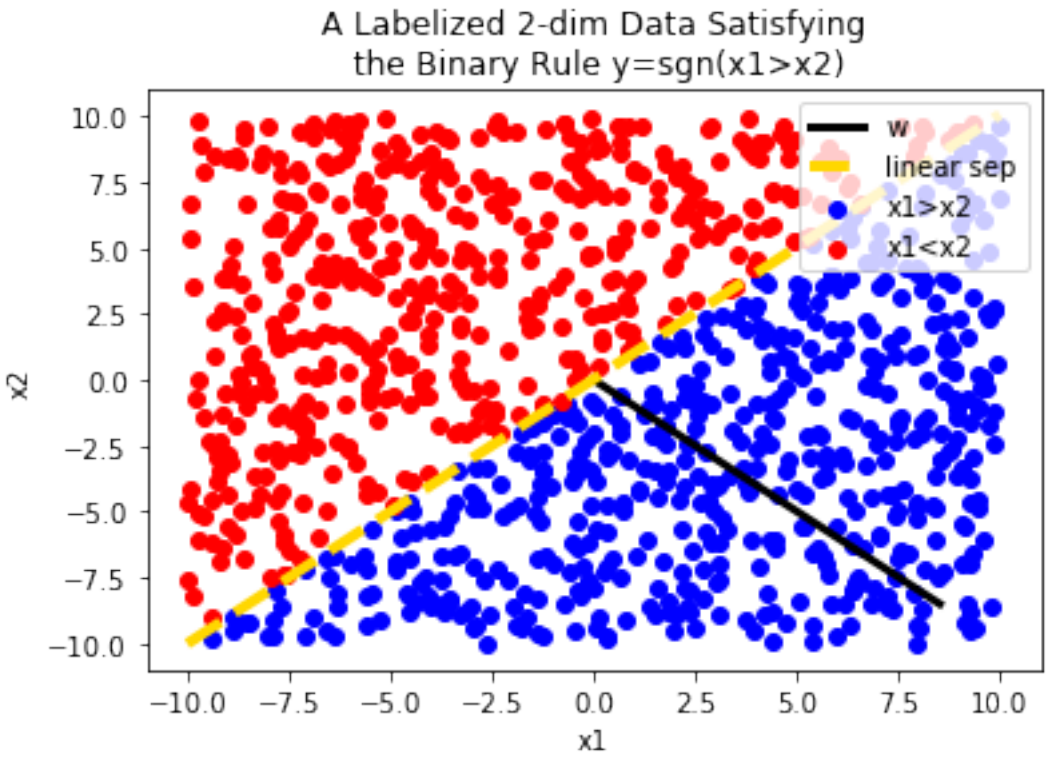
[0]:
```
weights = perceptron(mat,la,w_init=1)
```

Finding the seperator line by the orthogonal line to the w
We should find this line by the line equation

[0]:
```
sep_slope = -1/(weights[1]/weights[0])
x_base, y_base = (-10,-10)
y_of_x = (10-x_base)*sep_slope + y_base
```

Plotting with w and the seperator

[0]:
```
fig, ax = plt.subplots()
ax.scatter(blues[0,:],blues[1,:], c='blue', label = 'x1>x2')
ax.scatter(reds[0,:],reds[1,:], c='red', label = 'x1<x2')
ax.plot([0,(weights/5)[0]],[0,(weights/5)[1]],
        linewidth=3, c="black",label="w") # weights preserves its direction
ax.plot([x_base,10],[y_base,y_of_x], c="gold", linewidth=4, label="linear sep",␣
 ↪linestyle="--")
ax.set_xlim(-11,11)
ax.set_ylim(-11,11)
ax.title.set_text("A Labelized 2-dim Data Satisfying\n the Binary Rule␣
 ↪y=sgn(x1>x2)")
ax.set_xlabel("x1")
ax.set_ylabel("x2")
plt.legend()
plt.show()
```



Q4 Generate experiment df

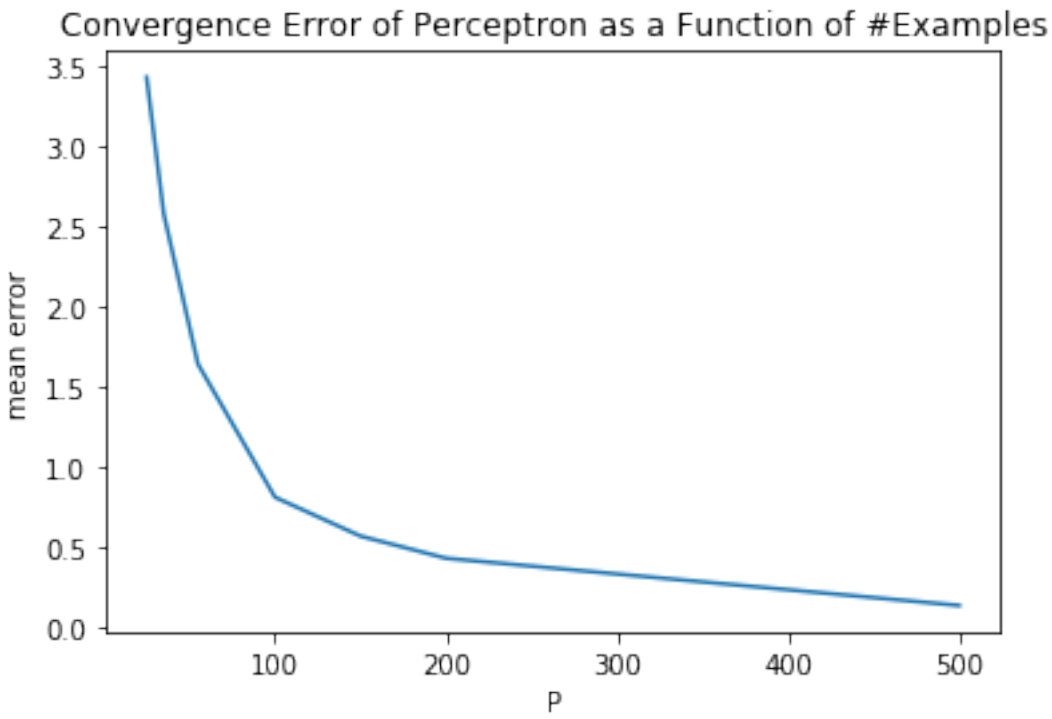
[0]:
```
W_opt = np.array([1,-1])
P = np.array([25, 35, 55, 100, 150, 200, 500])
experiment = pd.DataFrame(np.repeat(P,100), columns=['P'])
W_opt = np.array([1,-1]) # the optimal w in our case
experiment[["points","label"]] = pd.DataFrame(experiment['P'].apply(lambda p:␣
↪sample_two_dim_data(p)).tolist())
```

[0]:
```
def compute_error(points,label):
  w = perceptron(points,label,w_init=1)
  cos = (w @ W_opt)/ (np.linalg.norm(w) * np.linalg.norm(W_opt))
  error = abs(math.degrees(math.acos(cos)))
  return error
```

[0]:
```
experiment['error'] = experiment.apply(lambda row: compute_error(row["points"],␣
↪row["label"]), axis=1)
```

[0]:
```
means = []
for p in P:
  means.append(experiment.loc[experiment['P']==p, "error"].mean())
```

[0]:
```
plt.plot(P, means)
plt.title("Convergence Error of Perceptron as a Function of #Examples")
plt.xlabel("P")
plt.ylabel("mean error")
plt.show()
```


Convergence Error of Perceptron as a Function of #Examples
mean error
P
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
100
200
300
400
500