פעילות 6 - רגרסיה לינארית בסביבת Python

מקורות:

https://www.geeksforgeeks.org/linear-regression-python-implementation/

https://www.youtube.com/watch?v=szXbuO3bVRk

לפי ההגדרה רגרסיה ליניארית היא שיטה מתמטית למציאת הפרמטרים של הקשר בין משתנה בלתי תלוי X למשתנה תלוי Y, בהנחה שהקשר ביניהם ליניארי, כלומר מהצורה $y = mx + b$. (מקור: ויקיפדיה)

בפעילות זו נתרגל את היסודות המתמטיים של רגרסיה לינארית ויישומיה בשפת python. בפעילויות ההמשך נעשה שימוש בעקרונות הרגרסיה הליניארית כדי לכוון את המשקלים של רשתות נוירונים שנפתח בעתיד.

בפעילות זו נתמקד ברגרסיה לינארית בסיסית כדי למצוא את ערכי m ו- b של משוואת קו ישר:

$$y = mx + b$$

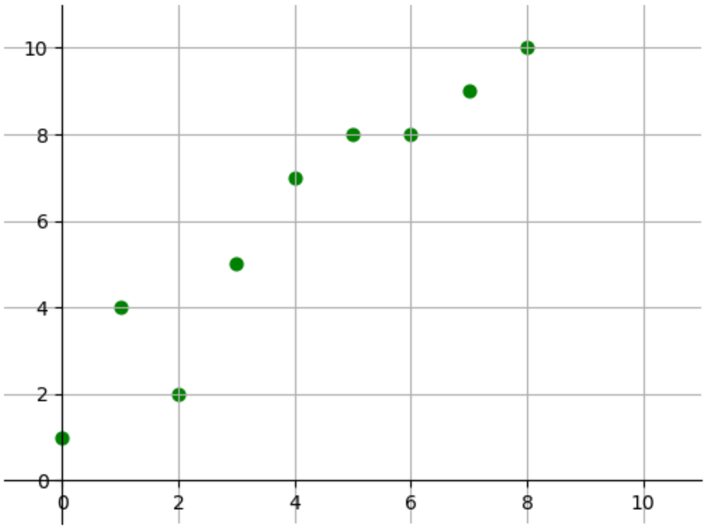
כלומר ההנחה הבסיסית כאן היא שאנו מנסים למצוא פונקציה לינארית שמנבאת את ערך של y בצורה מדויקת ככל האפשר כפונקציה של משתנה בלתי תלוי x.

כדי להבין את הנושא נגדיר 2 רשימות נתונים האחת מייצגת את הערכים של x והשנייה את הערכים של y. בפעילות זו ננתח את 2 הרשימות כדי ליצור מהם את משוואת הקו הישר המייצגת באופן אופטימלי את אוסף הנקודות שהגדרנו (כלומר כל 2 ערכים x ו- y מייצגים נקודה על מערכת צירים דו-מימדית).

נדגים זאת בקוד הבא:

```
import numpy as np

import matplotlib.pyplot as plt


ax = plt.gca()

ax.spines['left'].set_position('zero')

ax.spines['right'].set_color('none')

ax.spines['bottom'].set_position('zero')

ax.spines['top'].set_color('none')

plt.xlim([-1, 11])

plt.ylim([-1, 11])

plt.grid()


x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8])

y = np.array([1, 4, 2, 5, 7, 8, 8, 9, 10])


plt.scatter(x, y, color = "g", marker = "o", s = 40)

plt.show()
```

נקבל את גרף הנקודות הבא:



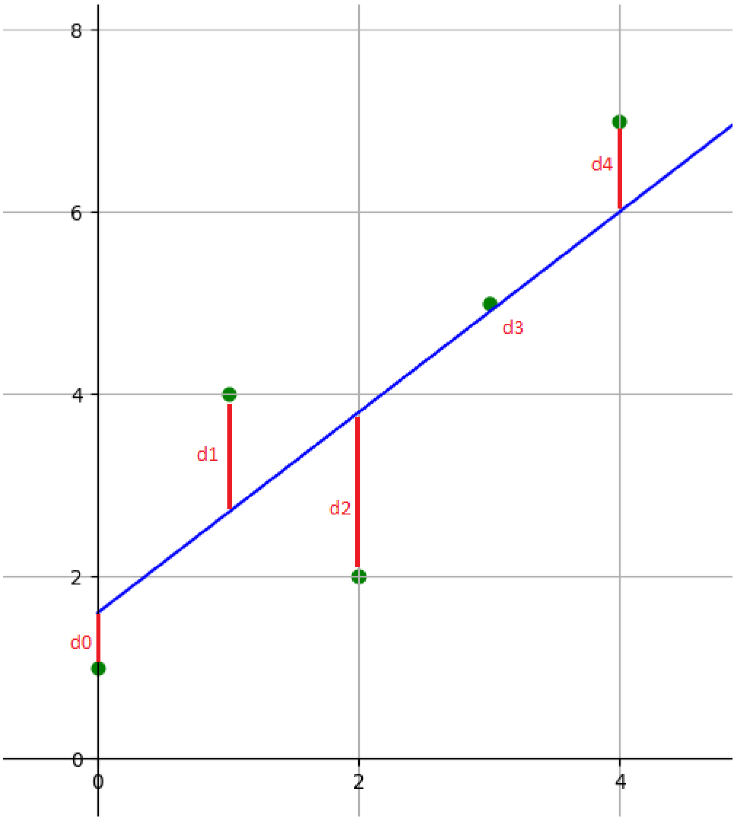
הגדרנו בקוד 2 וקטורים האחד בשם x השווה ל:

$$x = [x1, x2, x2......xn]$$

והשני וקטור בשם y שווה ל:

$$y = [y1, y2, y3..........yn]$$

כדי למצוא את קו המגמה המתאים ביותר למערך הנקודות נעזר בכלי המתמטי Ordinary least squares המאפשר לעשות סכום של כל הפרשי המרחקים בריבוע. נדגים זאת:



נציג את סכום ההפרשים בריבוע בעזרת הנוסחה הבאה:

$$(d0)^2 + (d1)^2 + (d2)^2 + (d3)^2 + ... + (dn)^2$$

המטרה בחישוב הנוסחה היא למצוא קו מגמה שבו תוצאת החישוב תהייה הערך הקטן ביותר האפשרי.

כדי להגיע לערך זה עלינו לדעת מה יהיה הערך של m ו- b (כלומר השיפוע של הקו וערך נקודת החיתוך שלו עם ציר y).

ניעזר בנוסחה הבאה כדי לחשב את ערכו של m האופטימלי, כזה שייתן לנו את השגיאה הקטנה ביותר האפשרית.

$$m = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

$$\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$$

נפרש את 3 הנוסחאות הבאות:

- הסימן $\sum_{i=1}^{n}$ מציין שמדובר בסכום כל האיברים החל מ-1 עד n.
- הסימנים $\bar{x}$ ו- $\bar{y}$ מציינים שמדובר בממוצע כל האיברים החל מ-1 עד n.

כדי לחשב את הערך של b נשתמש בנוסחה הבאה:

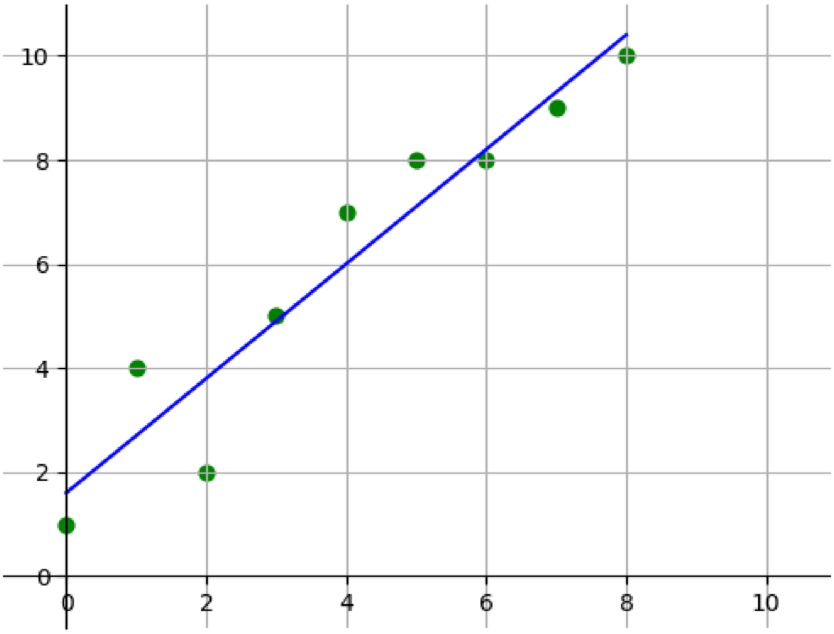
$$b = \bar{y} - m \cdot \bar{x}$$

נכתב את קוד התוכנית למציאת קו הרגרסיה הלינארית המתאימה ביותר למערך נתונים הכולל נקודות:

```
import numpy as np
import matplotlib.pyplot as plt


ax = plt.gca()
ax.spines['left'].set_position('zero')
ax.spines['right'].set_color('none')
ax.spines['bottom'].set_position('zero')
ax.spines['top'].set_color('none')
```

```
plt.xlim([-1, 11])
plt.ylim([-1, 11])
plt.grid()

x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8])
y = np.array([1, 4, 2, 5, 7, 8, 8, 9, 10])

avgx = np.mean(x)
avgy = np.mean(y)
m = (np.sum((x-avgx)*(y-avgy)))/(np.sum((x-avgx)*(x-avgx)))
b = avgy - m*avgx

print("m = ",m," b = ",b)
x_line = x
y_line = m*x + b
plt.plot(x_line, y_line, color = "b")
plt.scatter(x, y, color = "g", marker = "o", s = 40)
plt.show()
```

נקבל את הפלט הבא:



שימו לב לאופן שבו אנו ממירים את המשוואה הבאה לקוד בשפת Python:

$$m = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^{n}(x_i-\bar{x})^2}$$

```
avgx = np.mean(x)
avgy = np.mean(y)
m = (np.sum((x-avgx)*(y-avgy)))/(np.sum((x-avgx)*(x-avgx)))
```

הפעולה np.mean מחזירה את ממוצע כל האיברים שמקבלים כפרמטר לפעולה. כלומר המשתנים avgx ו- avgy מייצגים בקוד את המשתנים $\bar{x}$ ו- $\bar{y}$ בהתאמה.

הפעולה np.sum מחזירה את סכום כל האיברים שמקבלים כפרמטר לפעולה.

על פי אותו עיקרון הפונקציה $b = \bar{y} - m \cdot \bar{x}$ תמומש בקוד באופן הבא:

```
b = avgy - m*avgx
```

בפעילות הבא נלמד את יסודות מתמטיים ל- Gradient Descent ויישומיה בשפת python. כך שהנושאים רגרסיה ליניארית שלמדנו לממש בפעילות זו יחד עם Gradient Descent הם חלק מרכזי ביכולת ההבנה שלנו את נושא התכנות של רשתות נוירונים מלאכותיות ANN - Artificial Neural Network שנלמד בהמשך מדריך זה.

תנאי השימוש

תנאי השימוש במסמך זה הם לפי הסטנדרט הבא: