

מבוא ללמידת מכונה

למידת מכונה (Machine Learning) היא טכנולוגיה המאפשרת לפתור בעיות תוכנה, שקשה מאוד עד כדי כך שלא ניתן היה לפתור אותם בעזרת תכנות מסורתי הכולל משפטי if ולולאות for. טכנולוגיה זו מבוססת על היכולת לשפר באופן אוטומטי את ביצועי המחשב במשימות מורכבות. למידת מכונה משתמשת בשיטות מתמטיות/סטטיסטיות כדי לשפר את ביצועי האלגוריתם תוך כדי שימוש בנתונים העוברים דרך האלגוריתם.

להלן תרשים מלבנים המתאר עקרון פעולה של מחשב המבצע אלגוריתם תכנותי מסורתי המקבל במבוא נתונים ואלגוריתם. פלט המחשב יהיה נתונים מעובדים על פי האלגוריתם שסופק למחשב.



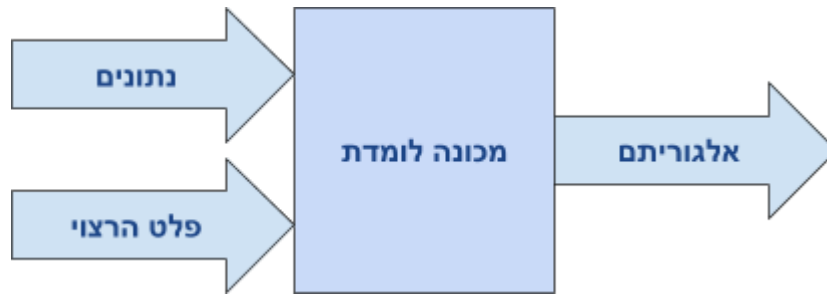
נדגים את עיקרון התכנות המסורתי כפי שמתואר בתרשים על ידי הקוד הבא:

```
import numpy as np
inputs = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
for i in inputs:
    out_arr = np.bitwise_and(i[0], i[1])
    print (i,"Output bitwise AND: ", out_arr)
```

ניתן לראות שהאלגוריתם שבשורות 3 עד 5 מבצע לולאה על נתוני המערך inputs. עבור כל נתון שמסופק לאלגוריתם דרך המערך מזמנים את הפעולה np.bitwise_and שמבצעת AND לוגי על 2 הערכים. בשלב האחרון מדפיסים את פלט האלגוריתם. פלט התוכנית יראה כך:

```
[0 0] Output bitwise AND: 0
[0 1] Output bitwise AND: 0
[1 0] Output bitwise AND: 0
[1 1] Output bitwise AND: 1
```

עקרון הפעולה של מכונה לומדת שונה בתכלית. האיור הבא מתאר את עקרון הפעולה של אלגוריתם המבוסס על מכונה לומדת:



למראית עין האיור נראה לא הגיוני. כיצד אוסף של נתונים ומידע על הפלט שאנו רוצים לקבל מסופקים במבוא מערכת מפיקים אלגוריתם במוצא? נדגים את העיקרון על ידי הקוד הבא:

```
import numpy as np

inputs = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
labels = np.array([0, 0, 0, 1])
perceptron = Perceptron(2)
perceptron.train(inputs, labels)

test = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
for i in test:
    print (i,"Output bitwise AND: ", perceptron.predict(i))
```

*כדי להדגיש את עקרון הפעולה של מערכת המבוסס מכונה לומדת. קוד התוכנית הבא אינו כולל את מימוש המחלקה Perceptron. הקוד המלא כולל מימוש מחלקה זו בליווי הסברים קיים בפעילות 9 שבמדריך זה.

ניתן לראות שהקוד הממש אלגוריתם ללמידת מכונה מחולק ל-2 שלבים. בשלב הראשון מסופקים למערכת 2 מערכים האחד בשם inputs הכולל את הנתונים והשני בשם labels הכולל את הפלט הרצוי מהמערכת. הפעולה perceptron.train מקבלת את 2 המערכים ונכנסת למצב למידה. בסוף התהליך הפלט יהיה אלגוריתם המסוגל לחשב פעולות AND בין 2 משתנים. בדיקת האלגוריתם תתבצע על ידי הגדרת מערך נוסף בשם test המסופק לפעולה perceptron.predict. פעולה זו תבדוק את נכונות האלגוריתם שנבנה על ידי המכונה בשלב ה- train. **מכאן שבנינו מכונה לומדת!** להלן דוגמה לפלט התוכנית:

```
[0 0] Output Machine Learning Algorithm: 0
[0 1] Output Machine Learning Algorithm: 0
[1 0] Output Machine Learning Algorithm: 0
[1 1] Output Machine Learning Algorithm: 1
```

ניתן להתרשם מיכולות המכונה ולספק לה סדרות שונות של נתונים על גבי אותו קוד בדיוק. בכך להתרשם שאותה המכונה מסוגלת ללמוד דברים שונים. לדוגמה נשנה את תוכן המערך labels לערכים שמייצגים לוגיקת OR:

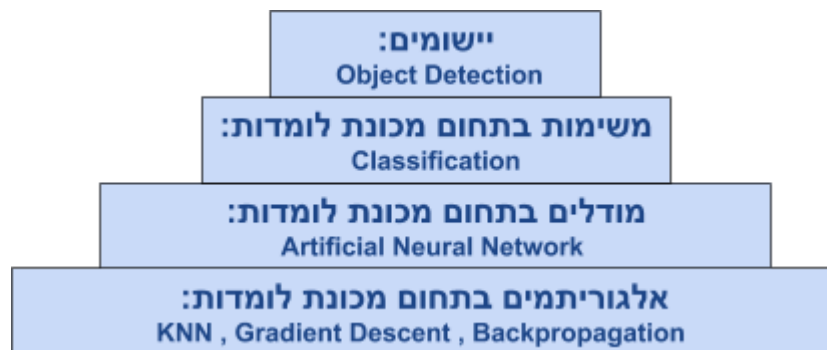
```
inputs = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
labels = np.array([0, 1, 1, 1])
```

```
perceptron = Perceptron(2)
perceptron.train(inputs, labels)
```

פלט התוכנית:

```
[0 0] Output Machine Learning Algorithm: 0
[0 1] Output Machine Learning Algorithm: 1
[1 0] Output Machine Learning Algorithm: 1
[1 1] Output Machine Learning Algorithm: 1
```

מכאן שאותה מכונה לומדת מסוגלת לייצר במוצא אלגוריתמים שונים בהתאם לנתונים שהתקבלו. בהמשך הפעילות נלמד שלא כל מכונה מסוגלת לייצר כל אלגוריתם, נלמד על אלגוריתמים כמו: KNN, Gradient Descent ו-Backpropagation, נלמד על מודלים למימוש מכונות לומדות כאשר המרכזי שבהם יהיה ANN – Artificial Neural Network שנקדיש לו חלק גדול במדריך זה. נעזר באלגוריתמים ומודלים במטרה לממש מכונות לומדות המסוגלות לבצע פעולות סיווג/מיון - Classification. לבסוף נאחד את הידע שצברנו כדי לכתוב יישומים לזיהוי מידע מתוך תמונות לדוגמה זיהוי מספרים ועצמים. נדגים את הדברים באיור הבא:



המדריך מחולק ל- 4 פרקים:
יסודות:

- פעילות 1 - תכנות גרפים תוך שימוש ב- Matplotlib
- פעילות 2 - מערכים תחת NumPy Arrays
- פעילות 3 - עבודה עם מטריצות תוך שימוש ב- NumPy
- פעילות 4 - פעולות (פונקציות) ומחלקות ב- Python
- פעילות 5 - ייצוג מידע ויזואלי במחשב
- פעילות 6 - רגרסיה לינארית בסביבת Python
- פעילות 7 - יסודות מתמטיים ל- Gradient Descent
- פעילות 8 - יישום רגרסיה לינארית על ידי Gradient Descent

תכנות מכונה לומדת צעד אחר צעד מהיסוד:

- פעילות 9 - יישום פרספטרון בודד
- פעילות 10 - מימוש שער XOR על ידי מספר נירונים
- פעילות 11 - פיתוח רשת נירונים מהיסוד
- פעילות 12 - אלגוריתם KNN ככלי לפיתוח מכונה לומדת

יישומי מכונה לומדת בשפת Python תחת windows10 :

- פעילות 13 - פיתוח מכונה לומדת לזיהוי ספרות בכתב יד תחת scikit-learn
- פעילות 14 - מימוש שער XOR על ידי מכונה לומדת תחת Keras

פעילות 15 - הפעלת The CIFAR-10 dataset בסביבת Python
פעילות 16 - תכנות מכונות לומדות תוך שימוש TensorFlow
פעילות 17 - גיבוי ושחזור מערך המשקלים של רשת נוירונים

יישומי מכונה לומדת בסביבת מיקרו-בקרים:
פעילות 18 - התקנת Raspberry PI להרצת אלגוריתם מכונה לומדת.
פעילות 19 - תכנות מכונה לומדת על Raspberry PI
פעילות 20 - תכנות רשת נוירונים על גבי מיקרו-בקר ממשפחת ESP32

תנאי השימוש

תנאי השימוש במסמך זה הם לפי הסטנדרט הבא:

You are free:

to Share – to copy, distribute and transmit the material
to Remix – to adapt the material

Under the following conditions:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

NonCommercial — You may not use the material for commercial purposes.

ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.