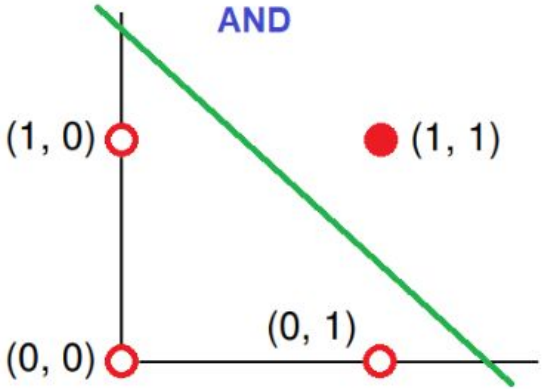
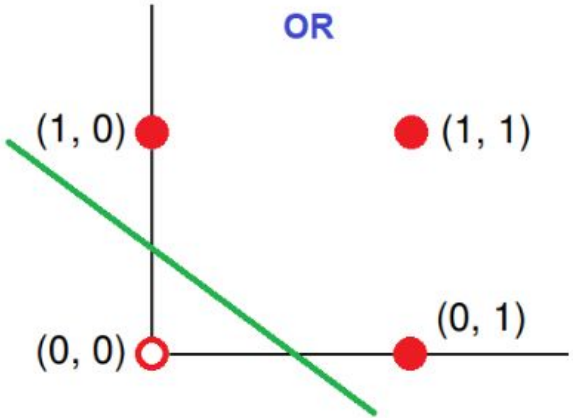
רשת נוירונים רדודה

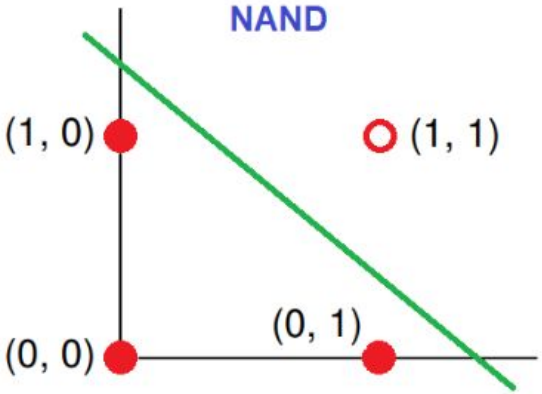
מימוש שער XOR על ידי מכונה לומדת

גדי הרמן

פרספטרון כמסווג לינארי



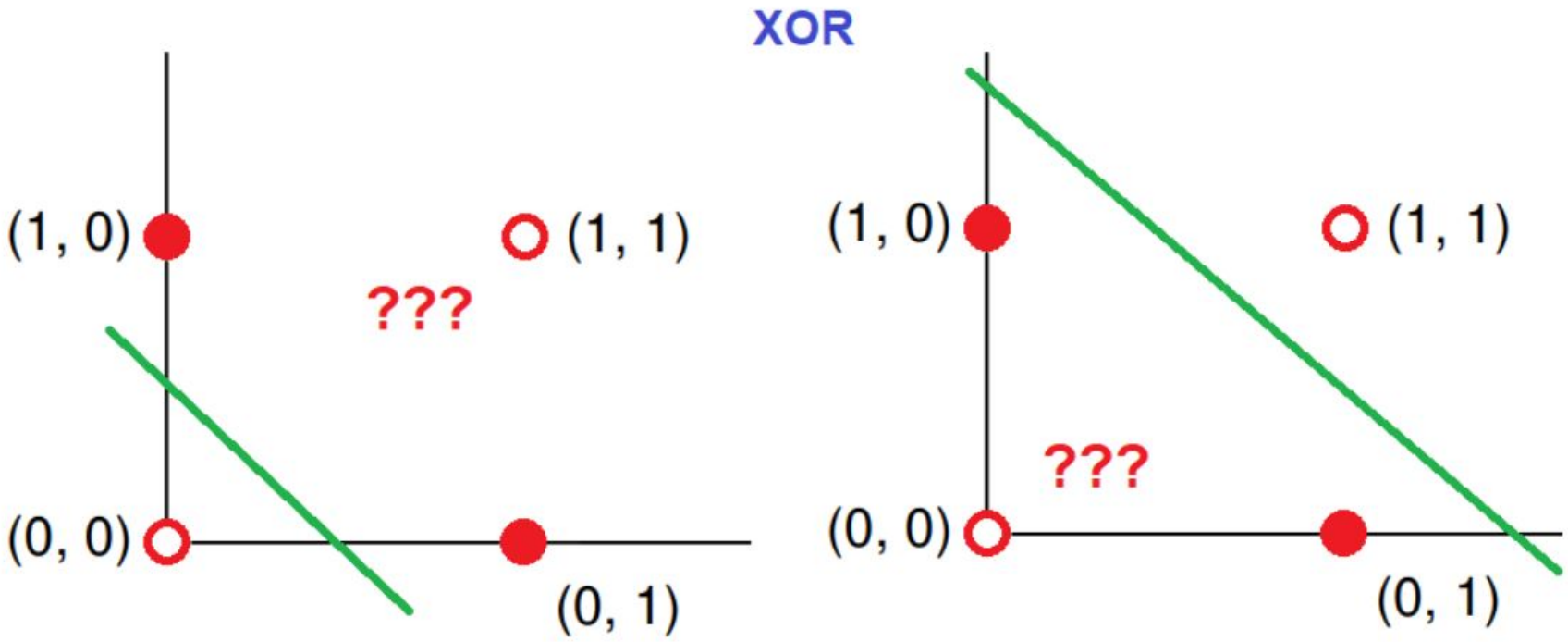




Input		Output
A	B	A xor B
0	0	0
0	1	1
1	0	1
1	1	0

```
[0 0] 1
[0 1] 1
[1 0] 1
[1 1] 0
```

פרספטרון כמסווג לינארי

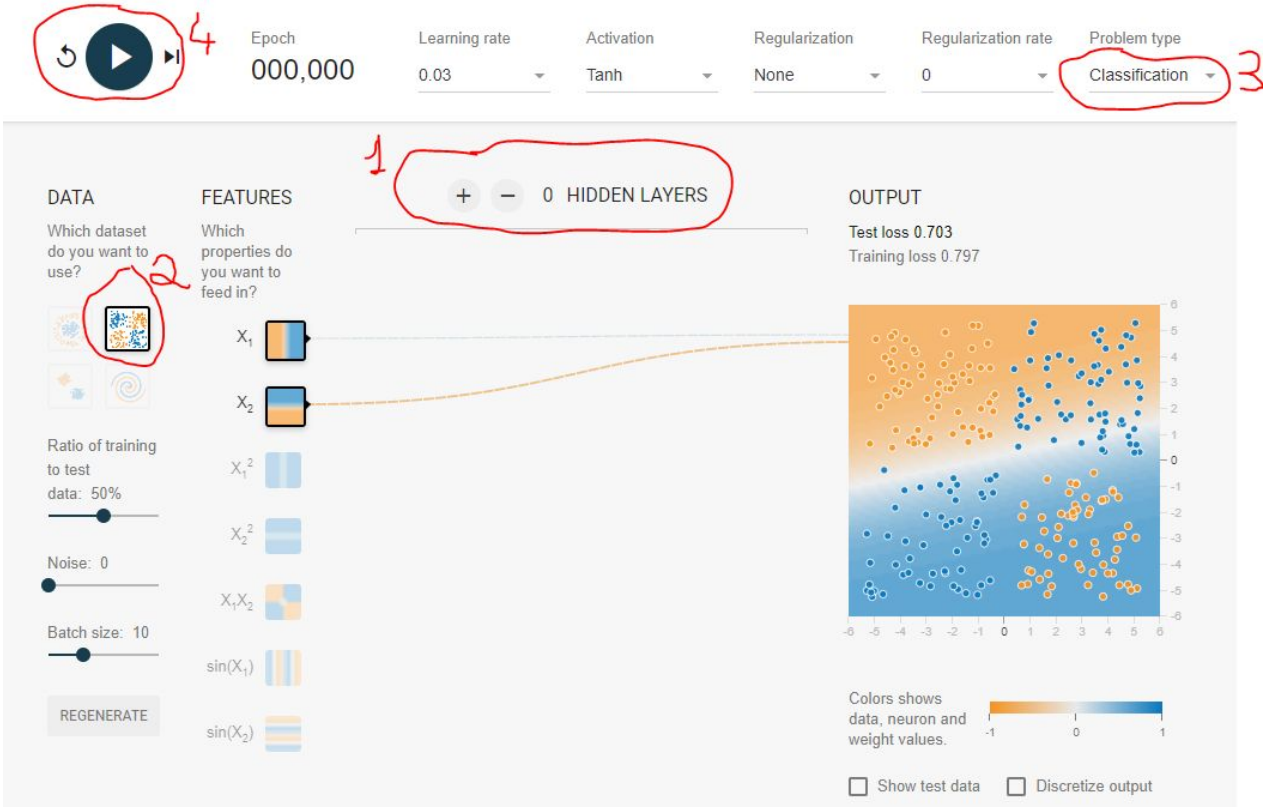


תרגול במליאה - פרספטרון כמסווג לינארי

כנסו באתר:

https://playground.tensorflow.org/

וממשו פרספטרון בודד באופן הבא:

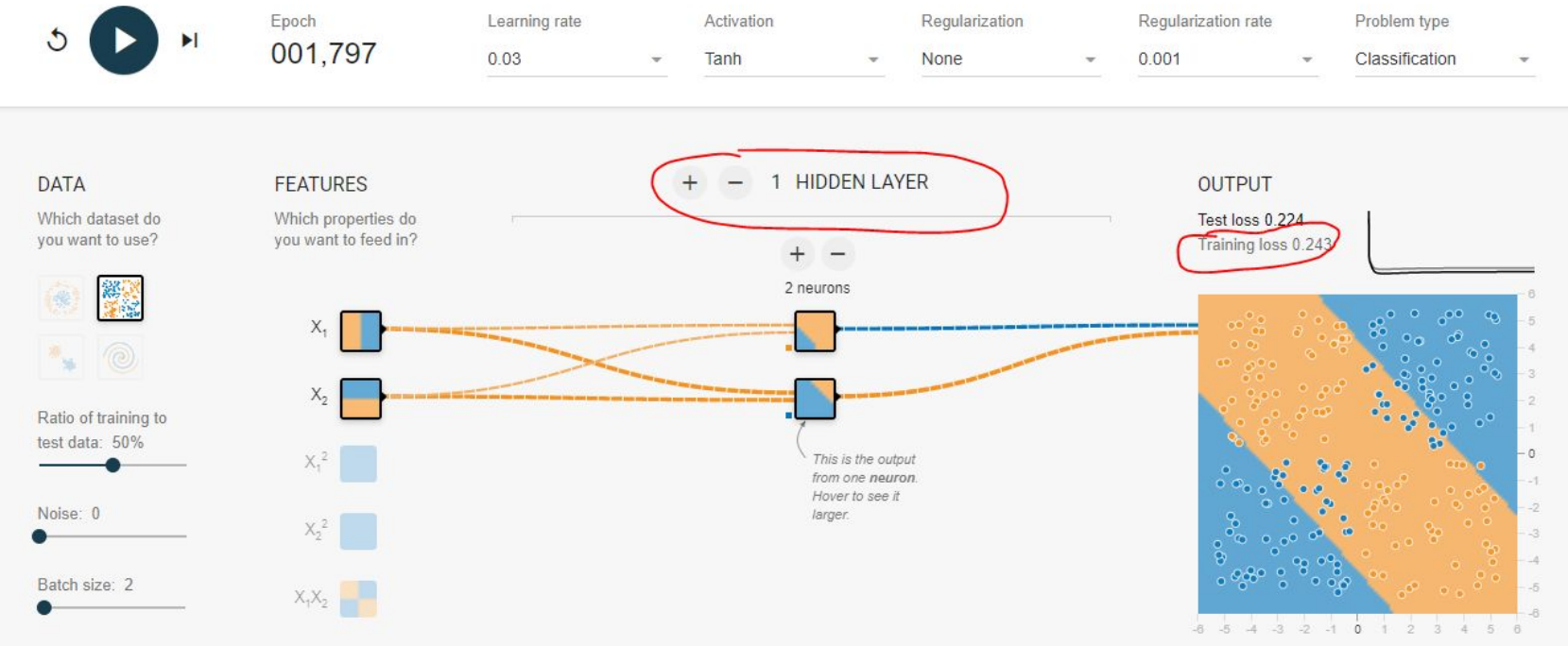


תרגול במליאה - פרספטרון כמסווג לינארי

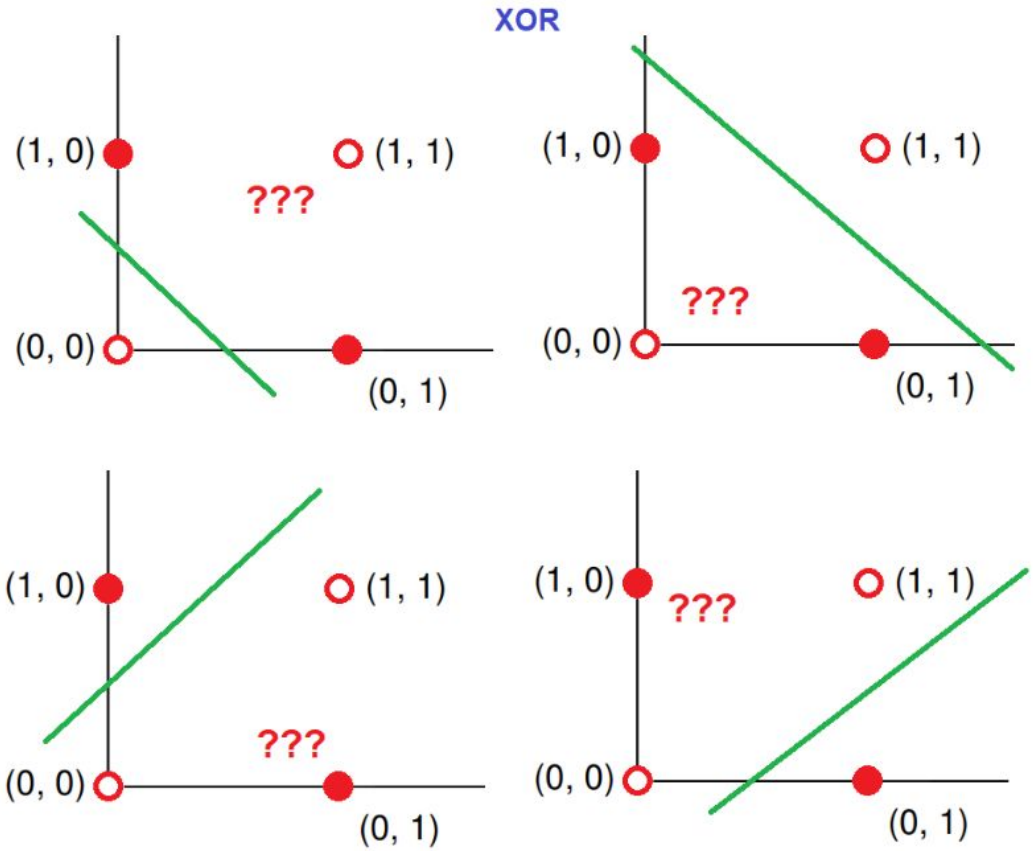
כנסו באתר:

https://playground.tensorflow.org/

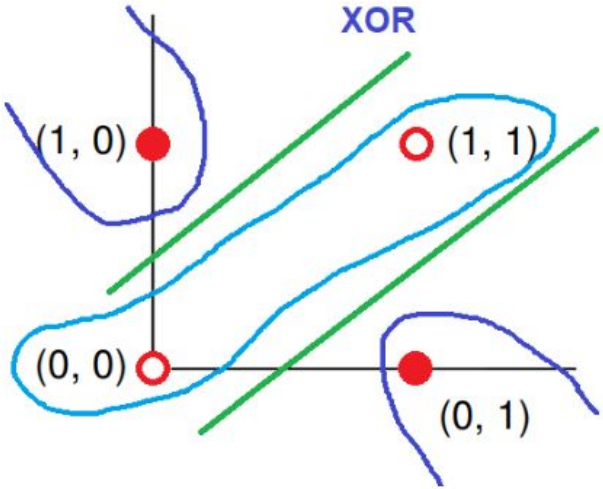
נוסיף 2 פרספטרונים נוספים כך שנקבל רשת הכולל 3 פרספטרונים האופן הבא:



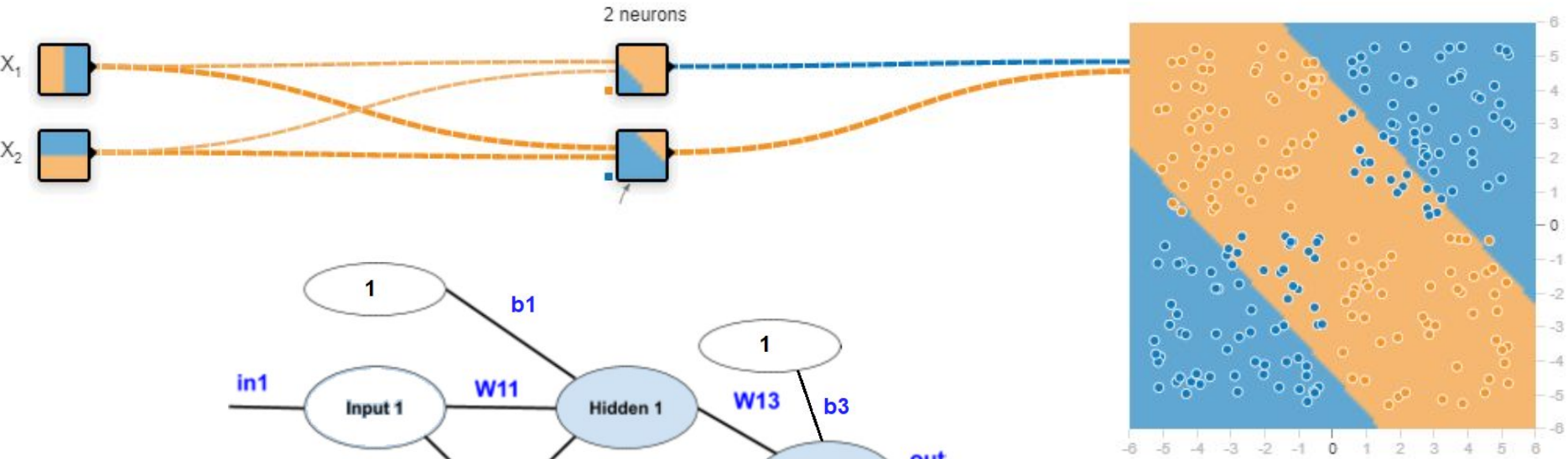
פרספטרון כמסווג לינארי

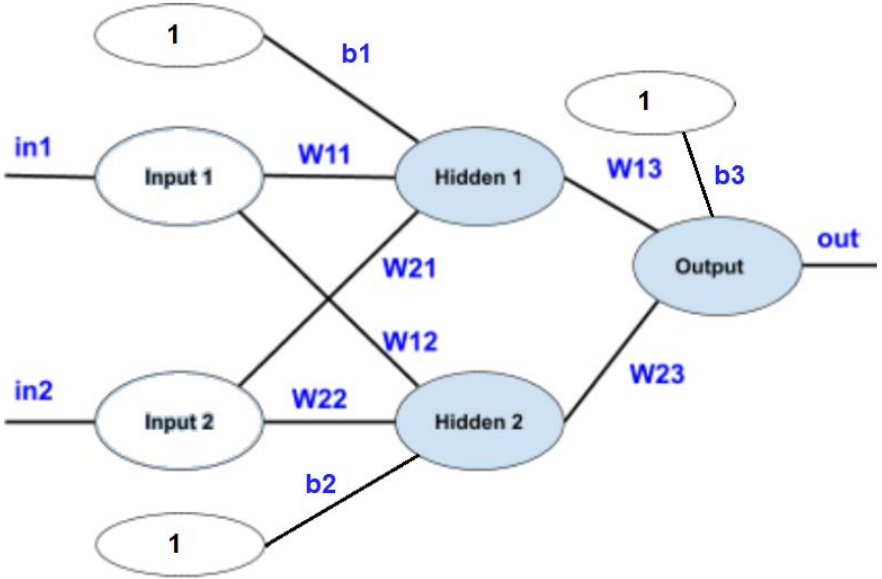


פתרון לכך יהיה מימוש רשת של מספר פרספטרונים במטרה לבנות מכונה המסוגלת ללמוד שער לוגי מסוג XOR.



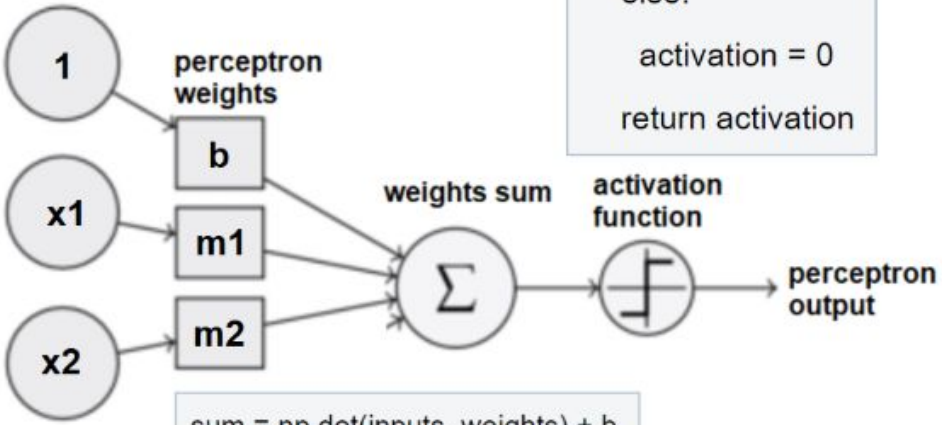
מבנה רשת נוירונים רדודה לסיווג XOR





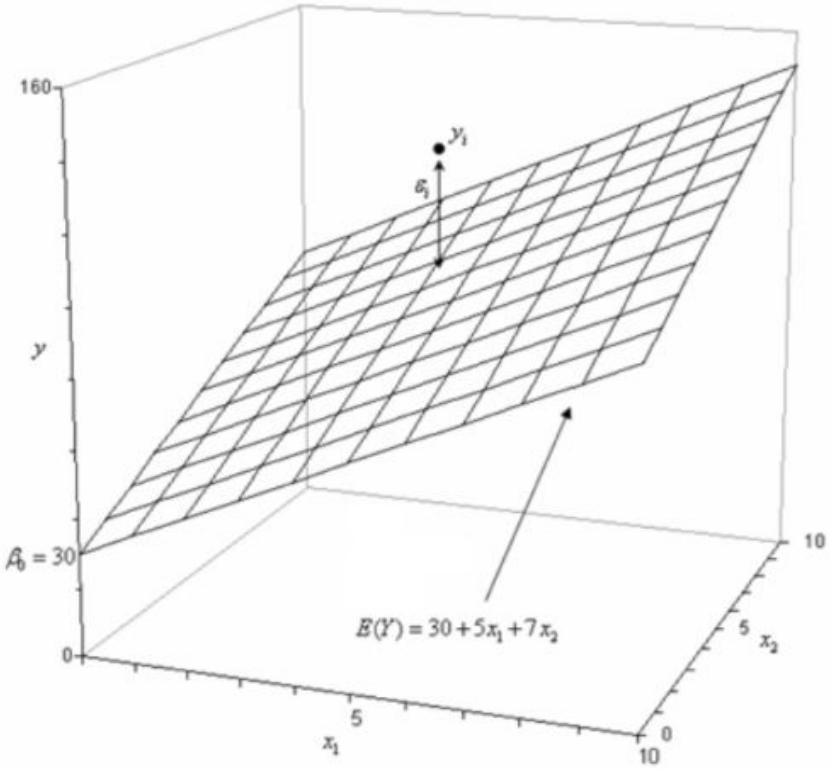
חזרה - פרספטרון כמסווג לינארי הכולל 2 מבואות

```
def Activation(s):
    if s > 0:
      activation = 1
    else:
      activation = 0
    return activation
```

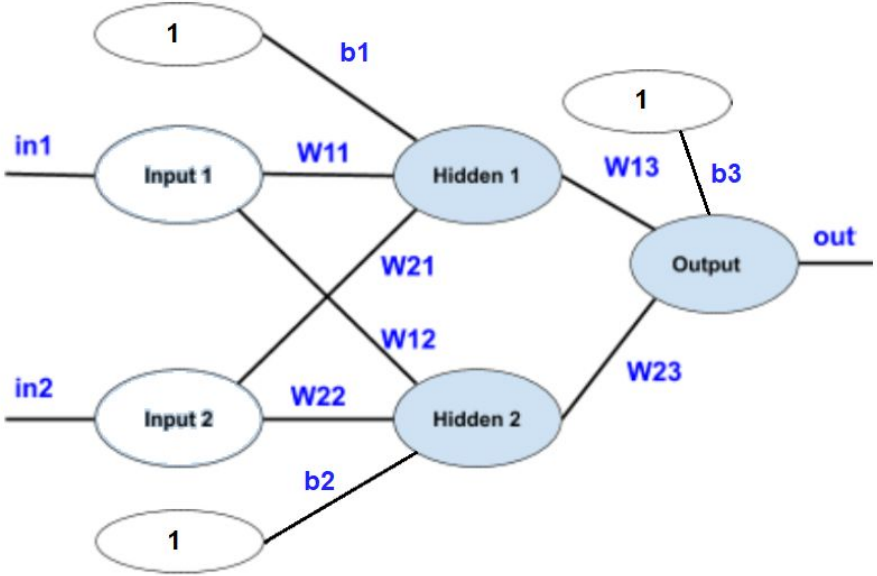


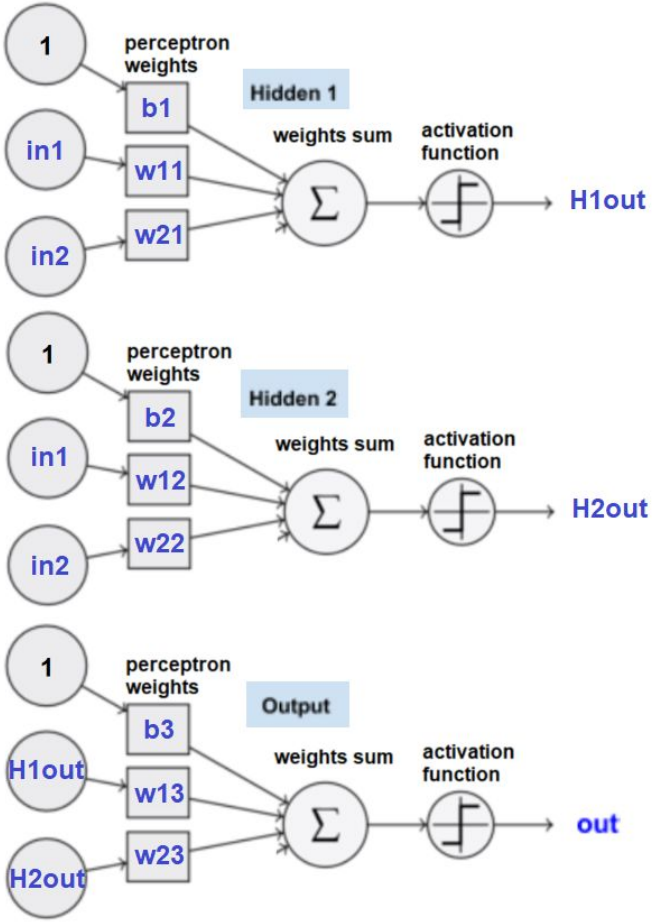

```
sum = np.dot(inputs, weights) + b
```

y=m1x1+m2x2+b



מבנה רשת נוירונים רדודה לסיווג XOR





אלגוריתם עבור רשת נוירונים רדודה לסיווג XOR

תיאור כללי של האלגוריתם ליישום רשת נוירונים רדודה:

1. נאתחל את כל הפרמטרים, כלומר ערכי m ו- b של אחד מהנוירונים, **בערכים אקראיים** בתחום שבין אפס לאחד.
2. נחשב את **הפלט הכללי** של מוצא רשת הנוירונים.
3. נחשב את **השגיאה הכללית**. כלומר ההפרש בין הערך הרצוי במוצא (אפס או אחד) לבין הערך שבמוצא הרשת ברגע זה.
4. נחשב את הנגזרת של **פונקציית השגיאה עבור כל הרשת**.
5. על סמך הנגזרת **נשנה את הפרמטרים של נוירון המוצא (Output)** בהתאם לשגיאה הכללית.
6. **נשנה את הפרמטרים של 2 הנוירונים בשכבה הפנימית** Hidden1 , Hidden2.
7. **נחזור לבצע את כל התהליך** מסעיף 2 עד שהשגיאה תהיה מינימלית.

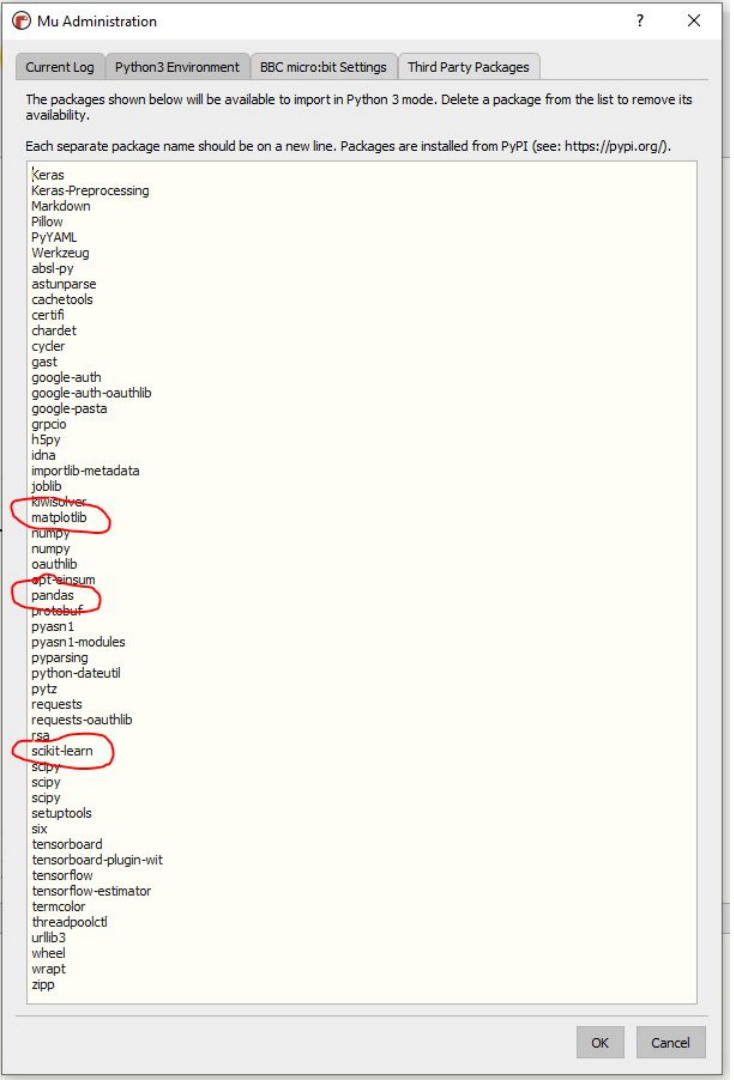
הספריה Scikit-learn

- הספרייה נכתבה כך שתתמוך בספריות משנה שכבר הכרנו כמו Numpy.
- הספרייה ברובה נכתבה בשפת Python כאשר חלקים ממנה נכתבו בשפת C כדי לתת ביצועים גבוהים.
- הספריה נכתבה במקור על ידי David Cournapeau כחלק מפרויקט של גוגל.
- ניתן לראות שימוש של הספריה כחלק ממערך המלצות השירים באפליקציה של Spotify כמו גם נעשה שימוש בספריה Scikit-learn באתר Booking.com כדי לספק לגולשים המלצות על מלונות ויעדים לטיול כמו גם איתור הונאות במערך ההזמנות של החברה.

Booking.com

Spotify®

התקנות




```
> python -m pip install -U pip
> python -m pip install -U matplotlib
> python -m pip install -U pandas
> python -m pip install -U scikit-learn
> python -m pip install -U mglearn
```

היכרות ראשונה על הספריה Scikit-learn

נתחיל בכתיבת קוד למכונה לומדת המבוססת על הספריה Scikit-learn כדי לממש שער XOR.

```
import numpy as np
import sklearn.neural_network

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

model = sklearn.neural_network.MLPClassifier(
                activation='logistic',
                max_iter=100,
                hidden_layer_sizes=(2,),
                solver='lbfgs')
model.fit(inputs, expected_output)
print('predictions:', model.predict(inputs))
```

היכרות ראשונה על הספריה Scikit-learn

```
model = sklearn.neural_network.MLPClassifier(
            activation='logistic',
            max_iter=100,
            hidden_layer_sizes=(2,),
            solver='lbfgs')
```

- פרמטר activation מגדיר את פונקציית התמסורת של פרספטרון. במקרה שלנו הגדרנו את הפונקציה כ- logistic כלומר סיגמואיד.
- פרמטר max_iter מגדיר מספר הפעמים שבה רשת הנוירונים תעבור על מערך נתוני האימון.
- פרמטר hidden_layer_sizes מגדיר את מערך השכבות הפנימיות של הרשת. במקרה שלנו הגדרנו שכבה בודדת אחת שבה 2 נוירונים.
- הפרמטר solver מגדיר האלגוריתם שבו רשת הנוירונים מבצעת חישובי משקלים. על פי המלצת המפתחים יש להשתמש במאפיין lbfgs כאשר עובדים עם מערך קטן של נתוני למידה.

https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

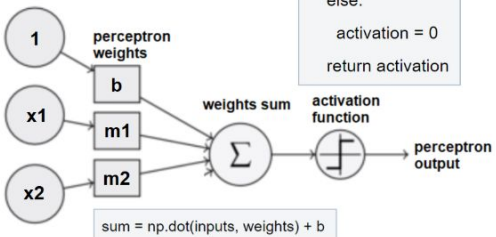
Activation function בחירת

activation : {*'identity', 'logistic', 'tanh', 'relu'*}, *default='relu'*

Activation function for the hidden layer.

- 'identity', no-op activation, useful to implement linear bottleneck, returns f(x) = x
- 'logistic', the logistic sigmoid function, returns f(x) = 1 / (1 + exp(-x)).
- 'tanh', the hyperbolic tan function, returns f(x) = tanh(x).
- 'relu', the rectified linear unit function, returns f(x) = max(0, x)

```
def Activation(s):
  if s > 0:
    activation = 1
  else:
    activation = 0
  return activation
```

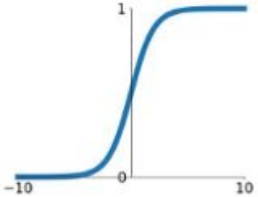


sum = np.dot(inputs, weights) + b

y=m1x1+m2x2+b

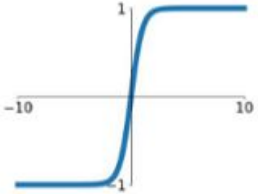
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



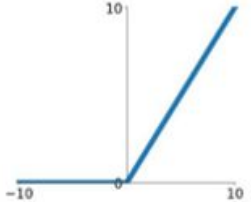
tanh

$$\tanh(x)$$



ReLU

$$\max(0, x)$$



https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

בחירת אלגוריתם gradient descent

solver : {'lbfgs', 'sgd', 'adam'}, default='adam'

The solver for weight optimization.

- 'lbfgs' is an optimizer in the family of quasi-Newton methods.
- 'sgd' refers to stochastic gradient descent.
- 'adam' refers to a stochastic gradient-based optimizer proposed by Kingma, Diederik, and Jimmy Ba

```
def train(self, inputs, labels):
  for _ in range(self.epochs):
    for i in range(len(inputs)):
      prd = self.predict(inputs[i])
      self.weights -= (prd - labels[i]) * inputs[i] * self.learningRate
      self.bios -= (prd - labels[i]) * self.learningRate
```

$$m = m - err \cdot x \cdot Learning\ Rate$$

$$b = b - err \cdot Learning\ Rate$$

https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

מימוש שער XOR על ידי
נוירון בודד (האם זה אפשרי?
אם כן למה שלא ננסה לכתוב
אחד כזה?)