Perceptrons and Exclusive Or (XOR)

Elsayed Issa

00/00/0000

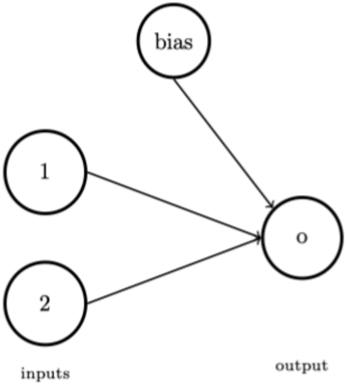
Table of Contents

Quiz

This quiz on last time and this time.

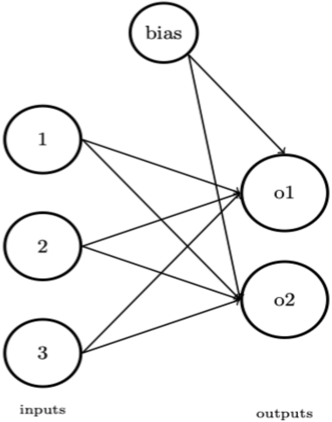
Perceptron

Two input nodes, a single output node, and a bias node.



Perceptron

Three input nodes, two output nodes, and a bias node.



Perceptron

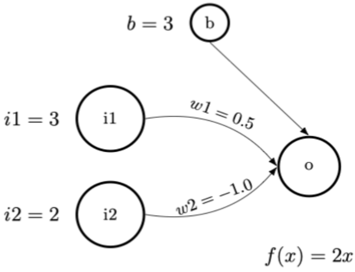
Recall

$$o(x) = f(\sum_{i=1}^{n} w_i x_i)$$

Let's add bias

$$o(x) = y = f(b + \sum_{i=1}^{n} w_i x_i)$$

1. x is a vector of n input values.
2. y is the number of output values and b is the bias value.
3. x_i and w_i are the input and weight values.
4. f is a linear function to apply.



$$o(x) = y = f(b + \sum_{i=1}^{n} w_i x_i) = ...$$

Practice: Solution

$$o(x) = y = f(b + \sum_{i=1}^{n} w_i x_i) = 2 \cdot (3 + 3 \cdot 0.5 - 2 \cdot 1.0) = 10.5$$

Practice: Solution

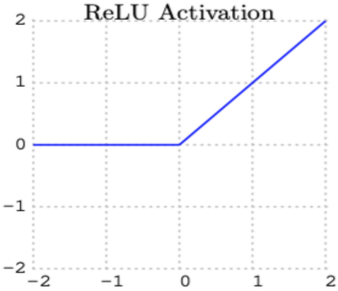

```
import numpy as np

class Perceptron:
    def __init__(self, x, w, b):
        self.x = x
        self.w = w
        self.b = b
    def some_function(self, i):
        return 2 * i
    def linear_function(self, i):
        return 1 if i > 0 else 0
    def perc(self):
        return (self.b + sum(self.x*self.w))
    def fit_some_function(self):
        return self.some_function(self.perc())
    def fit_linear_function(self):
        return self.linear_function(self.perc())
```

Practice: Solution


```python
import numpy as np
from perc import Perceptron

### Applying Some function ###
# inputs
x = np.array([3,2])
# weights
w = np.array([0.5, -1.0])
# bias
b = 3

p = Perceptron(x,w,b)
p.fit_some_function()
# 10.5
```

Rectified Linear Unit (ReLU)


```python
import numpy as np
from perc import Perceptron

### Applying Linear function ###
# inputs
x = np.array([3,2])
# weights
w = np.array([0.5, -1.0])
# bias
b = 3

p = Perceptron(x,w,b)
p.fit_linear_function()
# 10.5
```



$$g(z) = \max(0,z)$$

1. Perceptrons are basically an implementation of **linear regression**.
2. With a binary classifier, they are equivalent to **logistic** regression.
3. The problem with them is **linear separability** or **Exclusive Or (XOR)**
4. Exclusive **or** is true if precisely one of the disjuncts is true.
5. Logical **or** is linearly separable, but exclusive **or** is not.

a	b	$(a \vee b)$
T	T	T
T	F	T
F	T	T
F	F	F

a	b	$(a$ XOR $b)$
T	T	F
T	F	T
F	T	T
F	F	F

Given T is 1 and F is 0:

a	b	$(a \vee b)$
T	T	T
T	F	T
F	T	T
F	F	F

a	b	$(a \text{ XOR } b)$
T	T	F
T	F	T
F	T	T
F	F	F

a	b	$(a \vee b)$
1	1	1
1	0	1
0	1	1
0	0	0

a	b	$(a \text{ XOR } b)$
1	1	0
1	0	1
0	1	1
0	0	0

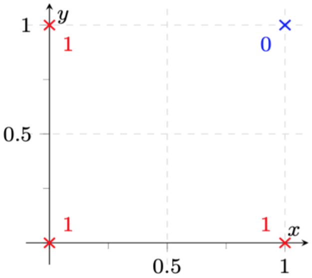
a	b	$(a \vee b)$
1	1	1
1	0	1
0	1	1
0	0	0

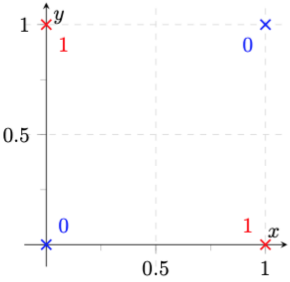
a	b	$(a \text{ XOR } b)$
1	1	0
1	0	1
0	1	1
0	0	0

$$y = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

$$y = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

$$y = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \qquad y = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$





Python demo

```python
import numpy as np
from sklearn.linear_model import Perceptron

x = np.array([[1,1],[1,0],[0,1],[0,0]])
y = np.array([1,1,1,0])
p = Perceptron(tol=1e-3)
p.fit(x,y)

print(f"actual output: {p.predict(x)}")

#Linearly separated output: [1 1 1 0]
```

Python demo


```python
import numpy as np
from sklearn.linear_model import Perceptron

x = np.array([[1,1],[1,0],[0,1],[0,0]])
y = np.array([0,1,1,0])
p = Perceptron(tol=1e-3)
p.fit(x,y)

print(f"actual output: {p.predict(x)}")

#Linearly non-separated output: [0 1 1 0]
```

Solving XOR with Linear Regression

Given T is 1 and F is 0:

a	b	(a XOR b)
T	T	F
T	F	T
F	T	T
F	F	F

a	b	(a XOR b)
1	1	0
1	0	1
0	1	1
0	0	0

we have:

$$X = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, y = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Solving XOR with Linear Regression

$$X = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, y = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Given X and y, what are the values for W and b given the following linear equation?

$$y = XW + b$$

Solving XOR with Linear Regression


```python
import numpy as np
from sklearn.linear_model import LinearRegression

# X.shape (2,4)
X = np.array([[1,1],[1,0],[0,1],[0,0]])
# Y.shape (4,1)
Y = np.array([[0],[1],[1],[0]])
# Create a LinearRegression model
model = LinearRegression()
# Fit the model to the data
model.fit(X, Y)
# Get the slope and intercept of the fitted model
slope = model.coef_[0]
# 0.0
intercept = model.intercept_
# 0.5
```

Solving XOR with Linear Regression

Apply linear equation:

$$y = XW + b$$

$$y = X \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}_{2\times 4} * W \begin{bmatrix} 0 \\ 0 \end{bmatrix}_{2\times 1} + 0.5$$

Apply linear equation:

$$y = XW + b$$

$$y = X \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}_{4\times 2} * W \begin{bmatrix} 0 \\ 0 \end{bmatrix}_{2\times 1} + 0.5$$

Solving XOR with Linear Regression

Apply linear equation:

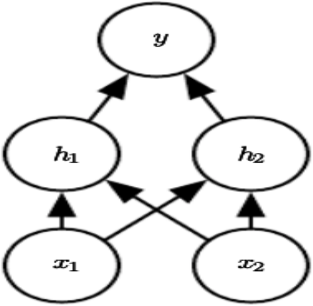
$$y = XW + b$$

$$y = X \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}_{4\times 2} * W \begin{bmatrix} 0 \\ 0 \end{bmatrix}_{2\times 1} + 0.5 = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}_{4\times 1}$$

XOR and Neural Networks

Neural nets can do non-separable cases.

$$f(X; W, c, w, b) = max(0, X^{T}W + c)w + b$$



Neural nets can do non-separable cases.

$$y \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = max(0, X \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{bmatrix} W + c) * w + b$$

Neural nets can do non-separable cases.

$$y \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = max(0, X \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{bmatrix} W = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} + c = \begin{bmatrix} 0 & -1 \end{bmatrix}) * w = \begin{bmatrix} 1 \\ -2 \end{bmatrix} + 0$$

Neural nets can do non-separable cases.

$$y\begin{bmatrix}0\\1\\1\\0\end{bmatrix} = max(0, \begin{bmatrix}0 & 0\\1 & 1\\1 & 1\\2 & 2\end{bmatrix} + \begin{bmatrix}0 & -1\end{bmatrix}) * \begin{bmatrix}1\\-2\end{bmatrix} + 0$$

Neural nets can do non-separable cases.

$$y \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = max(0, \begin{bmatrix} 0 & -1 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}) * \begin{bmatrix} 1 \\ -2 \end{bmatrix} + 0$$

XOR and Neural Networks

Neural nets can do non-separable cases.

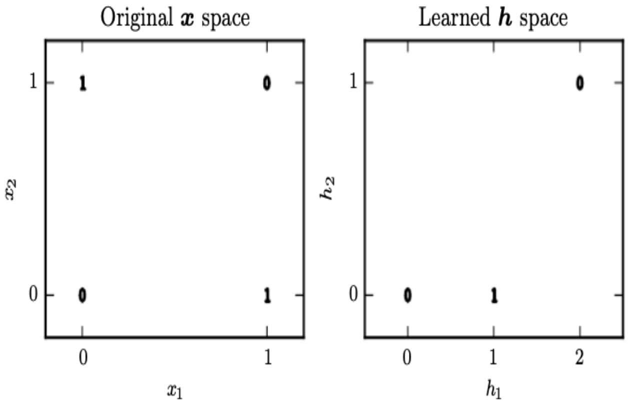
$$y \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ -2 \end{bmatrix} + 0$$

XOR and Neural Networks

Neural nets can do non-separable cases.

$$y \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} + 0$$

XOR and Neural Networks



References

Goodfellow, I., Bengio, Y., & Courville, A.(2016). Deep learning. MIT press. https://www.deeplearningbook.org