List 02 - Result

Tags: #fleeting #literature #permanent
Description:
Theme:

ID: 20250813131453

LIST – II RNA and LLM

1. Apply the perceptron learning algorithm to the case of the logical OR operator.

The inputs are given by x_0, x_1, and x_2, with weights $w_0(bias)$, w_1, and w_2, respectively, where y is the output that, according to the inputs, must equal the desired value d shown in the table below:

x_0	x_1	x_2	$x_1 \vee x_2$	d
1	1	1	1	1
1	1	0	1	1
1	0	1	1	1
1	0	0	0	0

The neural network to perform this learning task can be represented as shown in **Figure 1**.

The following initial values will be assigned to the weights and to the learning rate:

$$w_0 = 0, \quad w_1 = 0, \quad w_2 = 0 \quad \text{and} \quad \eta = 0.5 \tag{01}$$

The activation function (or transfer function) to be used is:

$$\varphi(v) = \begin{cases} 1, & \text{if } v > 0 \\ 0, & \text{if } v \leq 0 \end{cases} \tag{02}$$

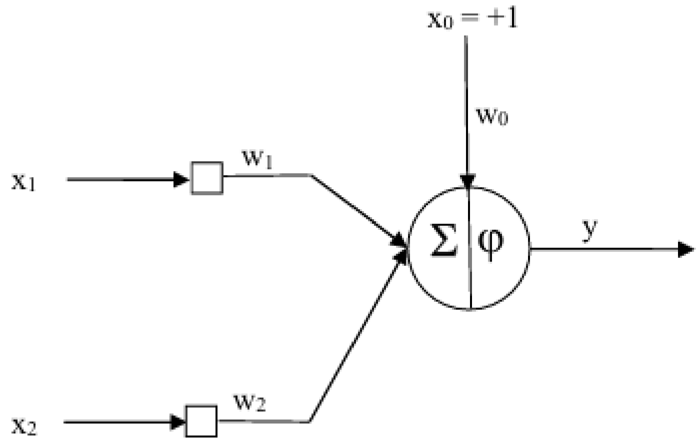

Figure 1 — Perceptron neural network for learning the OR logical operator.

After obtaining the results, construct a sequence of commands in **Python** to perform this training.

Resolution Delta Rule

First of all the input should be represented as a matrix where:

$$X_0 = [1,1,1,1], \quad X_1 = [1,1,0,0] \quad \textbf{and} \quad X_2 = [1,0,1,0] \tag{02}$$

And the weitgh matrix is given by

$$W_0 = [0,0,0,0], \quad W_1 = [0,0,0,0] \quad \textbf{and} \quad W_2 = [0,0,0,0] \tag{03}$$

so that, the function which will represent the perceptron is desing by:

$$U = X_0 \cdot W_0 + X_1 \cdot W_1 + X_2 \cdot W_2 \tag{04}$$

The output Y is represented by:

$$Y' = \varphi(U) \tag{05}$$

Wherein, :

$$\varphi(U) = \begin{cases} 1, & \text{if } u > 0 \\ 0, & \text{if } u \leq 0 \end{cases} \tag{06}$$

After that, the peceptron algorithm will adjust the output value using backpropagation and Delta Rule:

$$Y_{output} = Y - \eta \cdot \frac{\partial Y'}{\partial U} \cdot \frac{\partial U}{\partial W} \tag{07}$$

Resolution Rosenblatt

Inputs and desired outputs

There is **three inputs** per pattern:

$$\textbf{bias } x_0 = 1, \quad x_1, \quad x_2. \tag{03}$$

The dataset is:

$$\begin{array}{c|ccc|c} \text{pattern } i & x_0 & x_1 & x_2 & d \\ \hline 1 & 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 0 & 1 \\ 3 & 1 & 0 & 1 & 1 \\ 4 & 1 & 0 & 0 & 0 \end{array} \tag{04}$$

Weights and learning rate

Initial weights and learning rate are:

$$\boxed{w_0 = w_1 = w_2 = 0, \quad \eta = 0.5} \tag{05}$$

Activation (step) function

$$\varphi(v) = \begin{cases} 1, & v > 0, \\ 0, & v \le 0, \end{cases} \qquad \text{with } v = w_0 x_0 + w_1 x_1 + w_2 x_2 = w^\top x. \tag{06}$$

(Heaviside step as given.)

Algebraic notation

Training pattern as a 3-vector

$$x^{(i)} = [\, x_0, x_1, x_2 \,]^\top \quad \textit{with} \quad x_0 = 1. \tag{07}$$

The weight vector is:

$$w = [\, w_0, w_1, w_2 \,]. \tag{08}$$

Output is:

$$y^{(i)} = \varphi(v^{(i)}). \tag{09}$$

So write:

$$v^{(i)} = w^\top x^{(i)} = w_0 x_0^{(i)} + w_1 x_1^{(i)} + w_2 x_2^{(i)}, \qquad y^{(i)} = \varphi\Big(v^{(i)}\Big). \tag{10}$$

Learning rule

$$\boxed{w \;\leftarrow\; w \;+\; \eta\, (d^{(i)} - y^{(i)})\, x^{(i)}} \tag{11}$$

Interpretation by cases for each pattern $x^{(i)}$:

- If d$^{(i)} = 1$ and $y^{(i)} = 0$ (a **false negative**), then $w \leftarrow w + \eta\, x^{(i)}$ (push v upward).
- If $d^{(i)} = 0$ and $y^{(i)} = 1$ (a **false positive**), then $w \leftarrow w - \eta\, x^{(i)}$ (push v downward).
- If $d^{(i)} = y^{(i)}$, no change.

Algebric Solution

Using the step activation and the learning update:

$$y = \varphi(v), \quad v = w^\top x, \qquad w \leftarrow w + \eta\,(d - y)\,x, \quad \eta = 0.5, \quad w^{(0)} = (0, 0, 0). \tag{12}$$

Dataset (bias $x_0 = 1$):

$$\begin{array}{c|ccc|c} i & x_0 & x_1 & x_2 & d \\ \hline 1 & 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 0 & 1 \\ 3 & 1 & 0 & 1 & 1 \\ 4 & 1 & 0 & 0 & 0 \end{array} \tag{13}$$

Epoch 1 (start with $w = (0, 0, 0)$)

1. Pattern $x = [1, 1, 1]$, $d = 1$
   $v = 0 \Rightarrow y = 0$. Error $e = d - y = 1$.
   Update $w \leftarrow (0, 0, 0) + 0.5 \cdot 1 \cdot [1, 1, 1] = (0.5, 0.5, 0.5)$.
2. Pattern $x = [1, 1, 0]$, $d = 1$
   $v = 0.5 + 0.5 = 1.0 \Rightarrow y = 1$. $e = 0$. No change.
3. Pattern $x = [1, 0, 1]$, $d = 1$
   $v = 0.5 + 0.5 = 1.0 \Rightarrow y = 1$. $e = 0$. No change.
4. Pattern $x = [1, 0, 0]$, $d = 0$
   $v = 0.5 \Rightarrow y = 1$. $e = -1$.
   Update $w \leftarrow (0.5, 0.5, 0.5) + 0.5 \cdot (-1) \cdot [1, 0, 0] = (0, 0.5, 0.5)$.

End of epoch 1: $w = (0,\ 0.5,\ 0.5)$.

Epoch 2 (check for convergence)

- $[1, 1, 1]$: $v = 1.0 \Rightarrow y = 1 = d$ (ok)
- $[1, 1, 0]$: $v = 0.5 \Rightarrow y = 1 = d$(ok)
- $[1, 0, 1]$: $v = 0.5 \Rightarrow y = 1 = d$ (ok)
- $[1, 0, 0]$: $v = 0 \Rightarrow y = 0 = d$ (ok)

No mistakes ⇒ **converged**.

Final result

$$\boxed{w^* = (w_0, w_1, w_2) = (0,\ 0.5,\ 0.5)}$$

Decision rule: $y = 1$ if $v = w^\top x = 0.5\,x_1 + 0.5\,x_2 > 0$, i.e. whenever $(x_1, x_2) \neq (0, 0)$. That is exactly **OR**.

```
class Perceptron:
    def __init__(self, epochs=20, learning_rate=0.01):
        self.epochs = epochs
        self.learning_rate = learning_rate
```

```python
        self.W = None

    def perceptron_fit(self, X_train, y_train):
        X = X_train
        y = y_train
        n_samples, n_features = X.shape


        self.W = np.random.randn(n_features, 1)

        for epoch in range(self.epochs):
            for i in range(n_samples):
                x = X[i].reshape(-1, 1)
                u = np.dot(x.T, self.W)
                y_pred = np.heaviside(u, 1)
                error = y[i] - y_pred
                self.W += self.learning_rate * error * x

        return self

    def predict(self, X):

        if len(X.shape) == 1:
            X = X.reshape(1, -1)

        # Calcula a predição
        u = np.dot(X, self.W)
        y_pred = np.heaviside(u, 0)
        return y_pred.flatten()
```

Citações Importantes

- "Citação 1" (Página X) - Contexto:
- "Citação 2" (Página Y) - Contexto:

Comentários Pessoais

- **Reflexões**:
- **Críticas**:
- **Aplicações Práticas**:

Conexões

- **Relacionadas diretamente**:

    - Nota Relacionada 1
    - Nota Relacionada 2
- **Contexto mais amplo**:
    - Nota de Categoria 1
    - Nota de Categoria 2