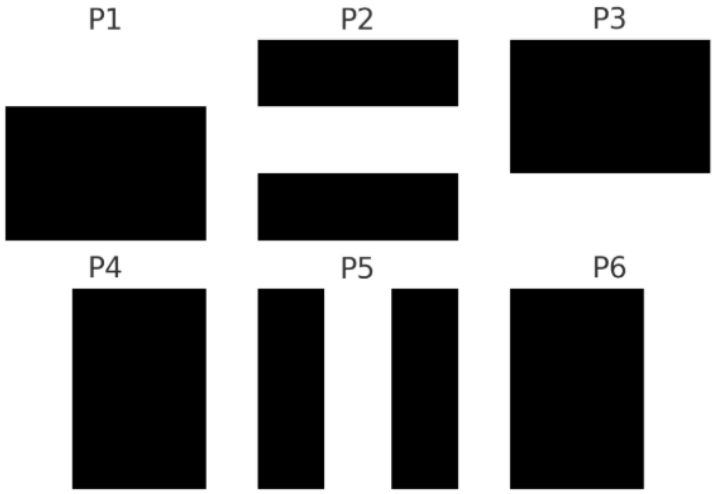
۱. تشخیص و جدا کردن الگوهای دودویی با پرسپترون تک‌لایه

الگوهای زیر را در نظر بگیرید (۱ = سفید، -1 = سیاه):

$$P_1 = \begin{bmatrix} 1 & 1 & 1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix}, \quad P_2 = \begin{bmatrix} -1 & -1 & -1 \\ 1 & 1 & 1 \\ -1 & -1 & -1 \end{bmatrix}, \quad P_3 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ 1 & 1 & 1 \end{bmatrix},$$

$$P_4 = \begin{bmatrix} 1 & -1 & -1 \\ 1 & -1 & -1 \\ 1 & -1 & -1 \end{bmatrix}, \quad P_5 = \begin{bmatrix} -1 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & -1 \end{bmatrix}, \quad P_6 = \begin{bmatrix} -1 & -1 & 1 \\ -1 & -1 & 1 \\ -1 & -1 & 1 \end{bmatrix}.$$

(آ) (الف (points نمایش تصویری هر الگو به صورت یک تصویر 3×3 دودویی:



شکل ۱: الگوهای P_1 تا P_6 به صورت پیکسل‌های سفید و سیاه

(ب) (ب (points طراحی یک پرسپترون تک‌لایه با یک نورون برای جدا کردن این الگوها:

- وزن‌ها و بایاس اولیه:

$$w_1 = w_2 = \cdots = w_9 = 0, \quad b = 0.$$

- تابع فعال‌سازی: $y = \operatorname{sign}\left(\mathbf{w}^\top \mathbf{x} + b\right) \in \{+1, -1\}$.
- قانون یادگیری پرسپترون:

$$w_i \leftarrow w_i + \eta\,(d - y)\,x_i, \quad b \leftarrow b + \eta\,(d - y),$$

که η نرخ یادگیری، d برچسب هدف و x_i مؤلفهٔ iام ورودی است.

برنامه ۱: پیاده‌سازی پرسپترون برای سوال ۸

```
# perceptron8.py
import numpy as np

patterns = np.array([
    [ 1,  1,  1, -1, -1, -1, -1, -1, -1],  # P1
    [-1, -1, -1,  1,  1,  1, -1, -1, -1],  # P2
```

```
    [-1, -1, -1, -1, -1, -1,  1,  1,  1],  # P3
    [ 1, -1, -1,  1, -1, -1,  1, -1, -1],  # P4
    [-1,  1, -1,  1, -1,  1, -1,  1, -1],  # P5
    [-1, -1,  1, -1, -1,  1, -1, -1,  1],  # P6
])

num_classes = patterns.shape[0]
num_features = patterns.shape[1]

targets = -np.ones((num_classes, num_classes))
for i in range(num_classes):
    targets[i, i] = 1

W = np.zeros((num_classes, num_features))
b = np.zeros(num_classes)
eta = 0.1

for i in range(num_classes):
    converged = False
    while not converged:
        converged = True
        for k, x in enumerate(patterns):
            d = targets[i, k]
            z = np.dot(W[i], x) + b[i]
            y = 1 if z >= 0 else -1
            if y != d:
                W[i] += eta * (d - y) * x
                b[i] += eta * (d - y)
                converged = False

for i in range(num_classes):
    print(f"Perceptron {i+1}: w = {W[i]}, b = {b[i]}")
```

در پایان آموزش تا همگرایی، وزن‌ها و بایاس نهایی به صورت زیر به دست می‌آیند:

$$\mathbf{w}_{\text{final}} = [\, w_1^*, w_2^*, \ldots, w_9^* \,], \quad b_{\text{final}} = b^*.$$