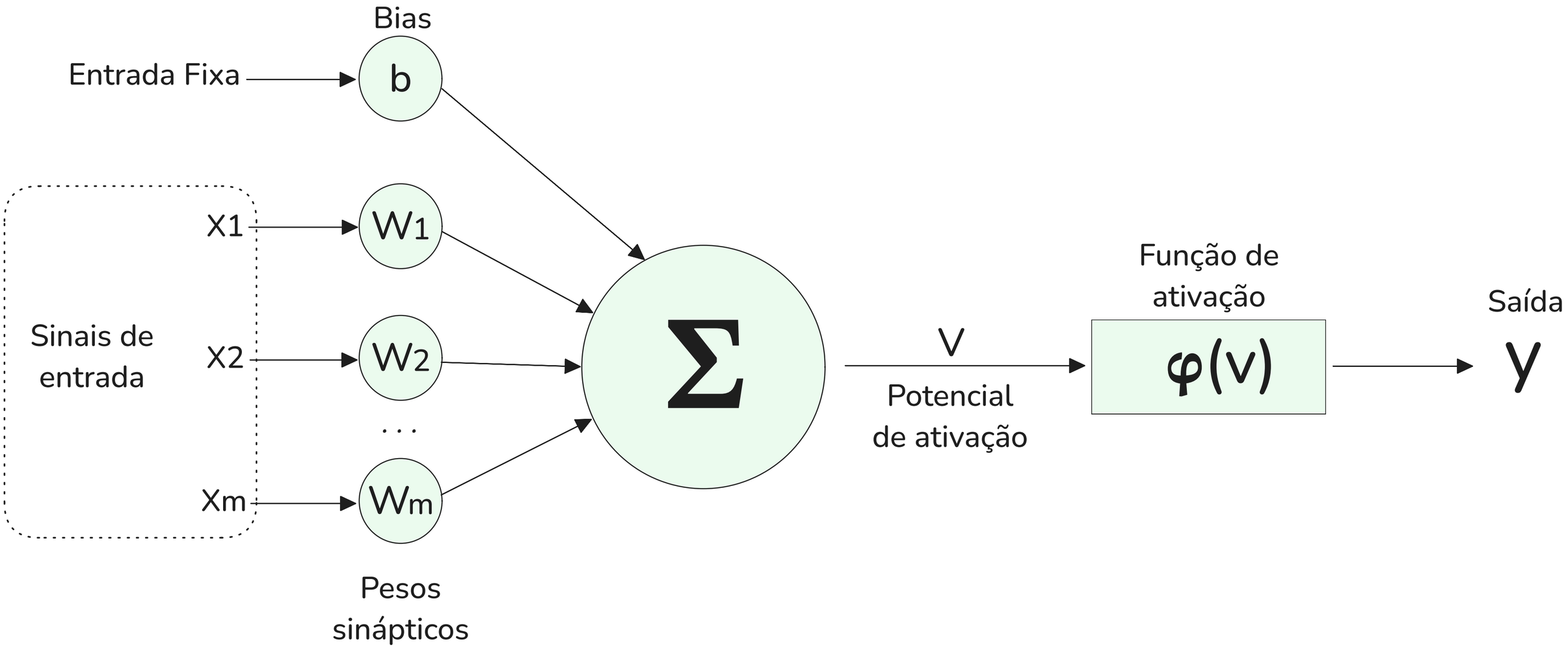
Perceptron de Rosenblatt (1958)



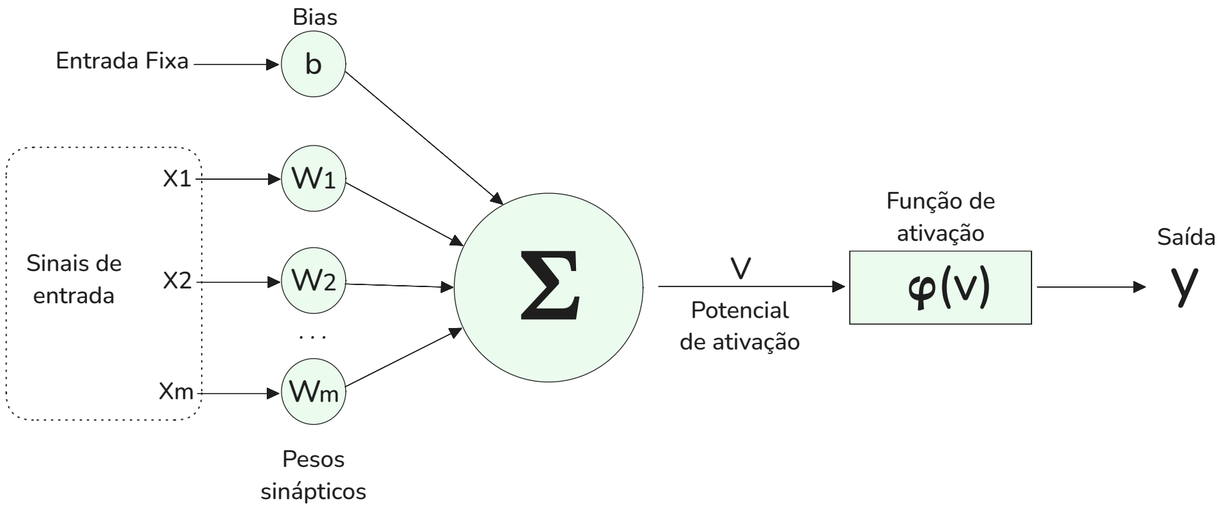
```python
# Função de ativação (função degrau)
def step_function(x):
    return np.where(x >= 0, 1, 0)
```

```python
# Classe do Perceptron
class Perceptron:
    def __init__(self, input_size, learning_rate=1.0, epochs=10):
        self.weights = np.zeros(input_size + 1)  # +1 para o bias
        self.learning_rate = learning_rate
        self.epochs = epochs

    def predict(self, x):
        x_with_bias = np.insert(x, 0, 1)  # Adiciona o bias
        weighted_sum = np.dot(self.weights, x_with_bias)
        return step_function(weighted_sum)

    def fit(self, X, y):
        for epoch in range(self.epochs):
            for xi, yi in zip(X, y):
                xi_with_bias = np.insert(xi, 0, 1)  # Adiciona o bias
                output = self.predict(xi)
                error = yi - output
                self.weights += self.learning_rate * error * xi_with_bias
```

Perceptron de Rosenblatt (1958)




```python
# Função de ativação (função degrau)
def step_function(x):
    return np.where(x >= 0, 1, 0)
```

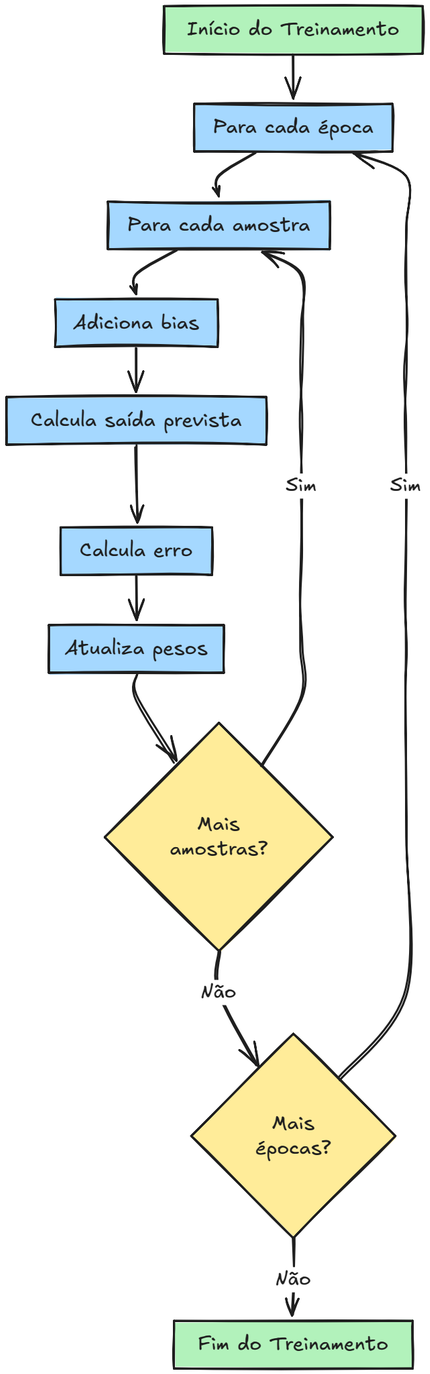
```python
# Classe do Perceptron
class Perceptron:
    def __init__(self, input_size, learning_rate=1.0, epochs=10):
        self.weights = np.zeros(input_size + 1)  # +1 para o bias
        self.learning_rate = learning_rate
        self.epochs = epochs

    def predict(self, x):
        x_with_bias = np.insert(x, 0, 1)  # Adiciona o bias
        weighted_sum = np.dot(self.weights, x_with_bias)
        return step_function(weighted_sum)

    def fit(self, X, y):
        for epoch in range(self.epochs):
            for xi, yi in zip(X, y):
                xi_with_bias = np.insert(xi, 0, 1)  # Adiciona o bias
                output = self.predict(xi)
                error = yi - output
                self.weights += self.learning_rate * error * xi_with_bias
```

Correspondência entre Diagrama de Rosenblatt e Código Python

Elemento da Figura	Correspondente no Código
X1, X2, ..., Xm (Sinais de entrada)	x (cada vetor de entrada do array X no código, por exemplo [1, 0])
W1, W2, ..., Wm (Pesos sinápticos)	self.weights[1:] (os pesos associados às entradas)
b (Bias / Entrada Fixa)	self.weights[0] (peso do bias; o valor constante 1 é inserido via np.insert(x, 0, 1))
∑ (Soma ponderada)	self.weights[0] (peso do bias; o valor constante 1 é inserido via np.insert(x, 0, 1))
v (Potencial de ativação)	weighted_sum (a soma ponderada entre pesos e entradas, incluindo o bias)
ϕ(v) (Função de ativação)	step_function(x) – a função degrau binária (0 ou 1)
y (Saída do neurônio)	output (retorno da função predict(x))


Início do Treinamento
Para cada época
Para cada amostra
Adiciona bias
Calcula saída prevista
Calcula erro
Atualiza pesos
Mais amostras?
Sim
Não
Mais épocas?
Sim
Não
Fim do Treinamento