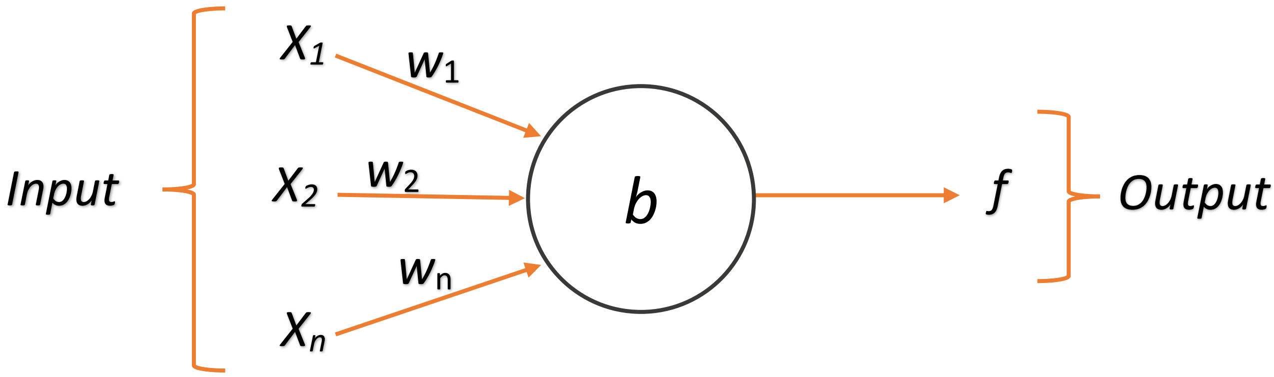
SINGLE-LAYER PERCEPTRON

1. Determine all the necessary ***x***, the weight for each *x* - $\boldsymbol{w_x}$ (Since the neuron state is bipolar, the weights can be set to values between -1 and 1) and ***b*** is the condition that will satisfy the Sign function, then:



where $\boldsymbol{w_n}$ = weights, ***b*** = bias (threshold)

$$f \begin{cases} 0 & \Sigma_j \, w_j \, x_j \le -b \\ 1 & \Sigma_j \, w_j \, x_j > -b \end{cases}$$

$$f \begin{cases} 0 & w * x + b \le 0 \\ 1 & \underbrace{w * x + b}_{\text{Activation function}} > 0 \end{cases}$$

2. Insert values into the formula $\boldsymbol{y = sign\,(w_o * x_0 + w_1 * x_1 + w_n * x_n)}$ and Calculate Neuron Output ***y***.
3. Update Weights: $\boldsymbol{w_i = w_i + n * (d-y) * x_i}$, where
   - $\boldsymbol{w_i}$ = weight to be updated
   - ***n*** is the learning rate (a small positive constant)
   - ***d*** is the desired output
   - ***y*** is the actual output
   - $\boldsymbol{x_i}$ is the input corresponding to the weight $\boldsymbol{w_i}$

4. Repeat steps 2-3 for multiple iterations until the network converge.

CONCLUSION

We have explored the foundational steps of training a single-layer perceptron for a specific function through the process, the perceptron learns by iteratively updating its weights $\boldsymbol{w}$ and bias $\boldsymbol{b}$ based on the perceptron learning rule, aiming to minimize the difference between the actual and target outputs. The learning rate plays important role in controlling the size of weight and bias updates, influencing the convergence of the perceptron. It's important to remember that this basic perceptron model has limits, such as its inability to solve problems that are not linearly separable. For more complex tasks, multilayer perceptron or other advanced neural network may be better or even required.