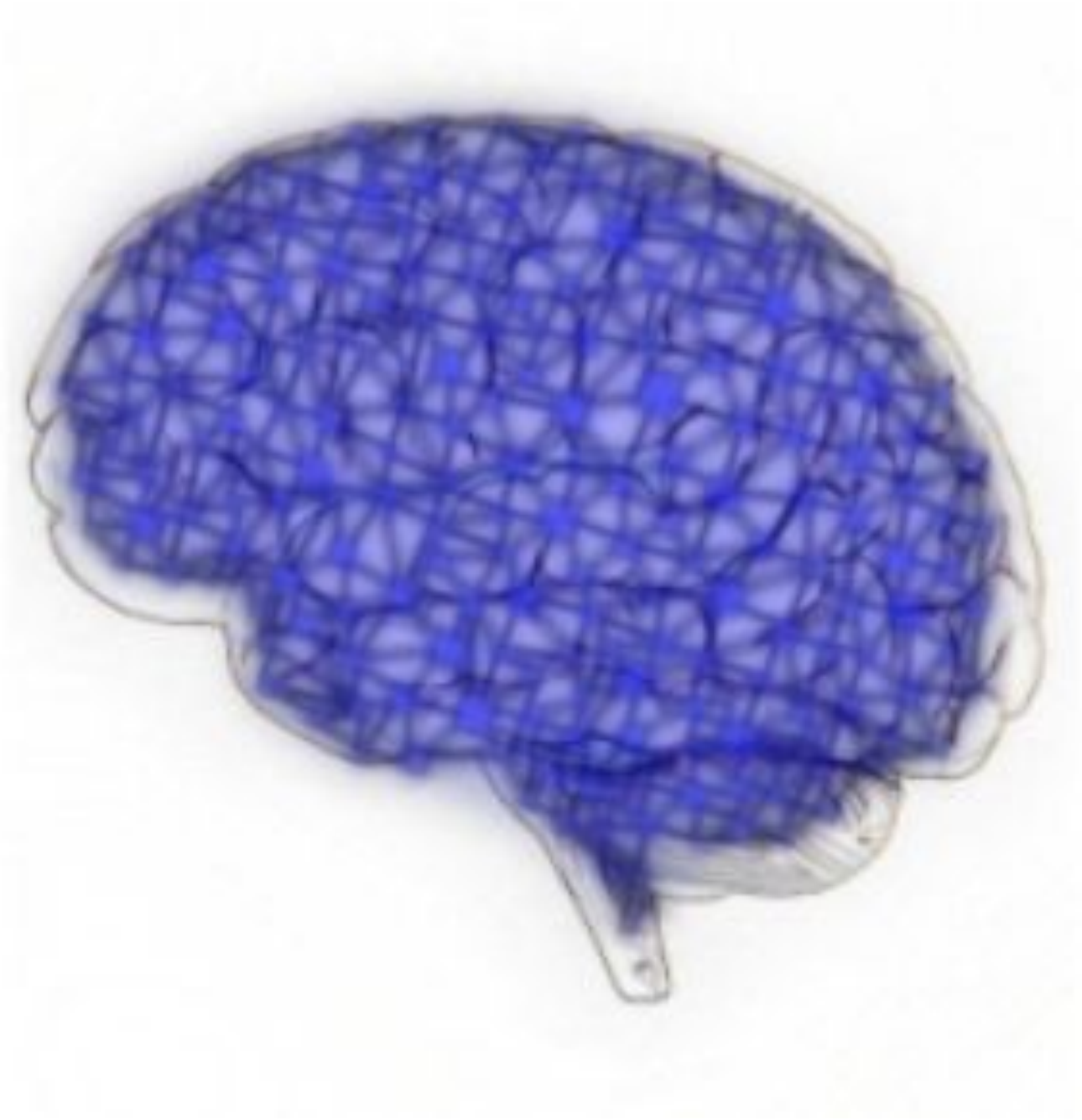
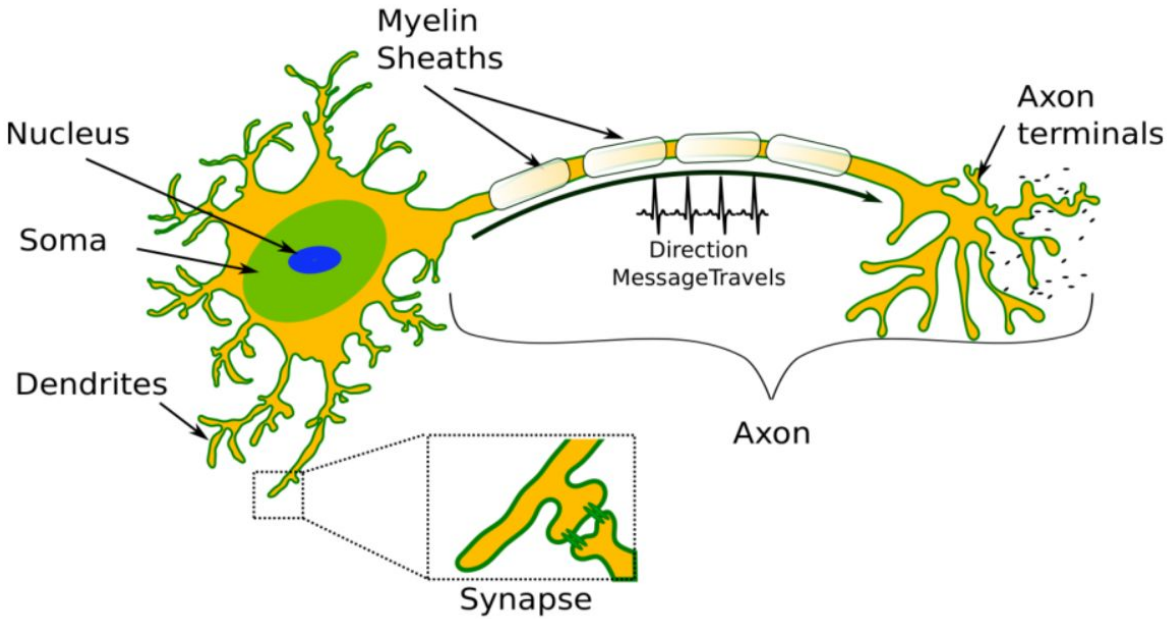


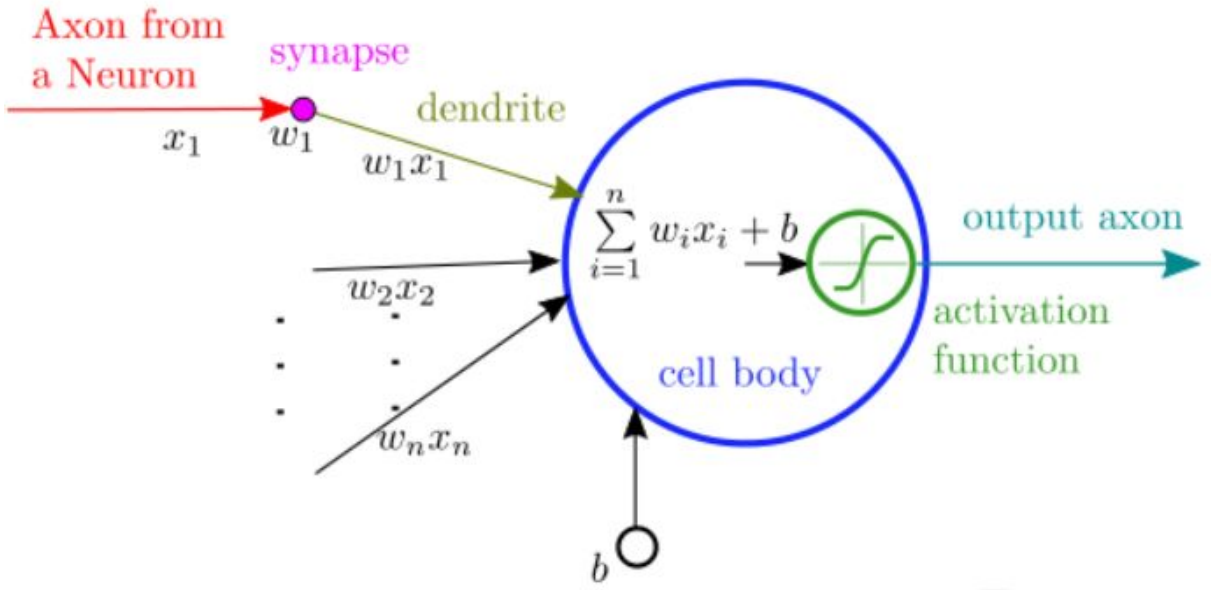
Artificial Neural Networks - Perceptron

PhD. Msc. David C. Baldears S.
PhD(s). Msc. Diego López Bernal

Artificial Neuron

A common neuron has inputs and if some threshold is pass it sends a signal (this is not completely true but was the original base)





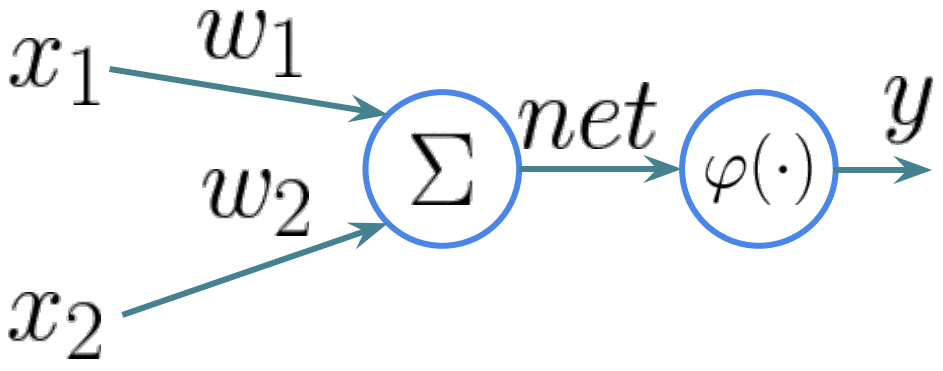
Perceptron

The perceptron is a single layer feed-forward neural network.

Frank Rosenblatt (1960's)

The output is a hyperplane (e.g. 2D Line)

$$net = w_1 x_1 + w_2 x_2$$
$$y = \varphi(net)$$

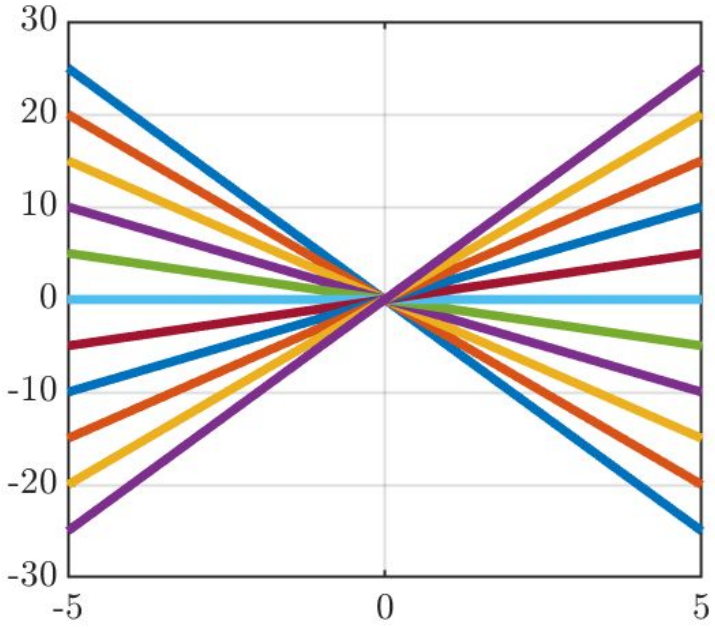


$$\varphi = \begin{cases} 1 \text{ if } net \geq 0 \\ 0 \text{ if } net < 0 \end{cases}$$

Bias

Yet, this hyperplane is centered at the origin

$$net = w_1 x_1 + w_2 x_2$$
$$y = \varphi(net)$$



How to move a line from the origin?

Bias

Yet, this hyperplane is centered at the origin

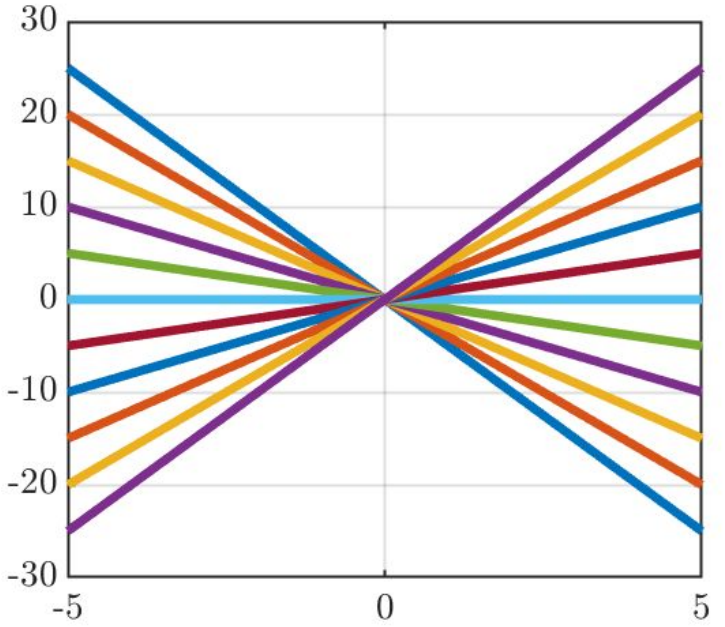
$$net = w_1 x_1 + w_2 x_2$$
$$y = \varphi(net)$$

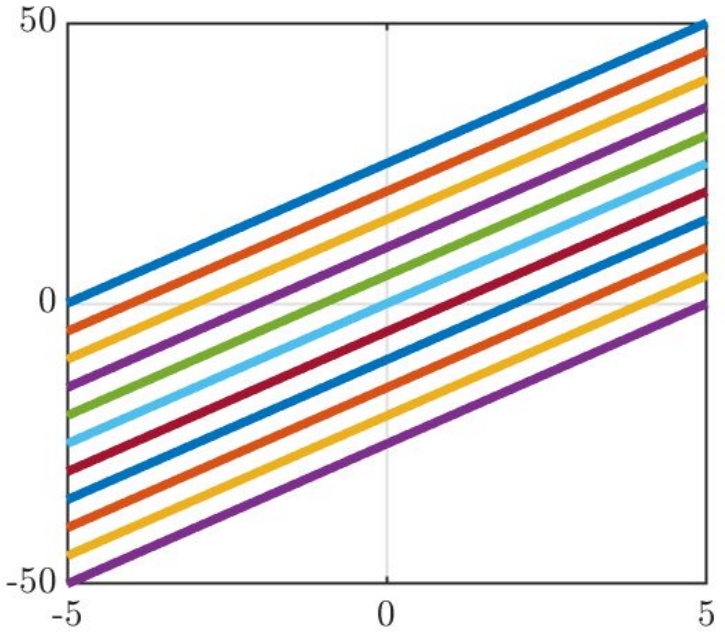
How to move a line from the origin?

Using the Bias

$$y = mx + b$$

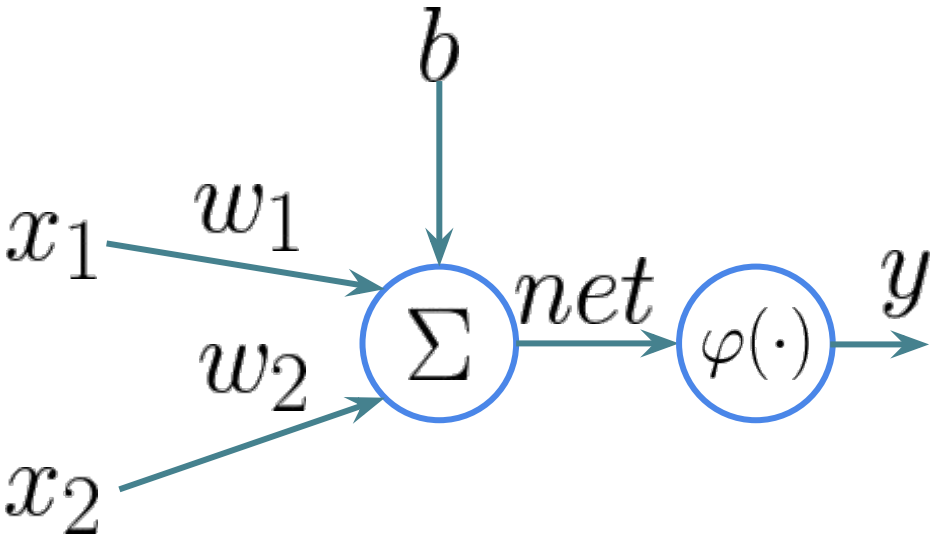
The bias is proportional to the offset of the plane from the origin





Perceptron

Adding the bias would help, yet, the bias has to be treated differently trainable



$$net = w_1 * x_1 + w_2 * x_2 + b$$

$$y = \varphi(net)$$

Unless???

$$\varphi = \begin{cases} 1 \text{ if } net \geq 0 \\ 0 \text{ if } net < 0 \end{cases}$$

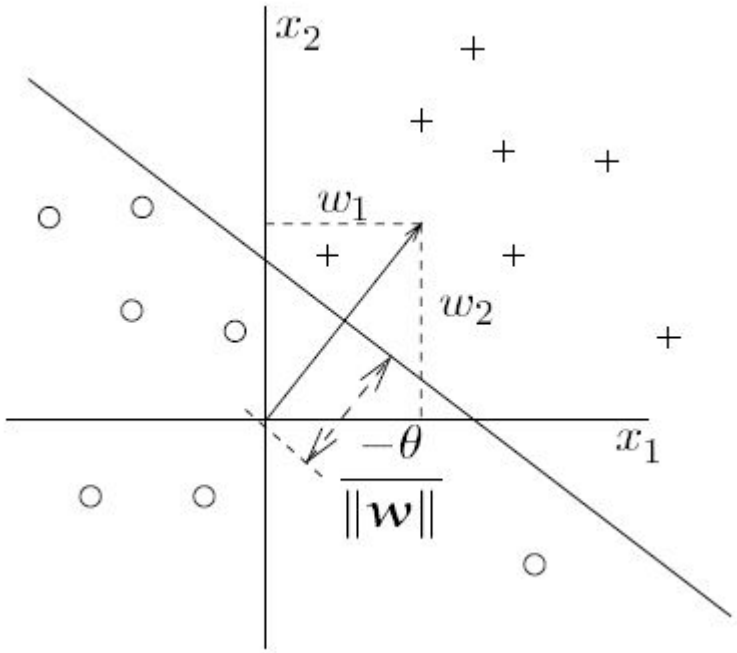
Linearly Separable

The bias is proportional to the offset of the plane from the origin

The weights determine the slope of the line

The weight vector is perpendicular to the plane

$$x_2 = -\frac{w_1}{w_2}x_1 - \frac{b}{w_2}$$



Learning Algorithm

- The main idea is to train the perceptron to classify inputs correctly
- Accomplished by adjusting the connecting weights and the bias
- Can only properly handle linearly separable sets
- Training has to adapt weights and bias using the inputs x’s and labels(targets) t’s
- Let, α, be the learning rate
- Initialize the weights and bias (randomly)

Learning Algorithm

- Set Desired output (target) $$t(n) = \begin{cases} +1 \text{ if } x(n) \text{ class } 1 \\ -1 \text{ if } x(n) \text{ class } 2 \end{cases}$$

- Use inputs x's and labels(targets) t's, α, and weights

1. Select a random sample from training set as input
2. If classification is correct, do nothing
3. If classification is incorrect, modify the w using

$$w_i = w_i + \alpha t(n) x_i(n)$$

4. Repeat until the entire training set is classified correctly

Perceptron Simple Example

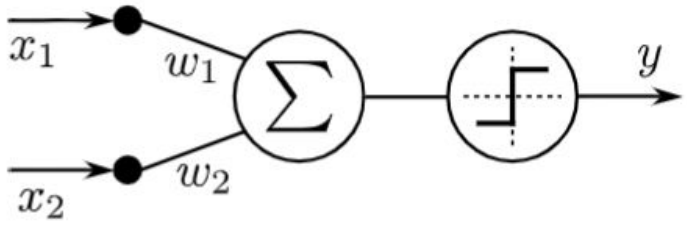
For simplification define

$$net(x) = \sum_{i=1}^{n} x_i \cdot w_i = x_1 \cdot w_1 + x_2 \cdot w_2 + ... + x_n \cdot w_n$$

$$\sigma(net) = \begin{cases} 1, & \text{if } net \geq \theta \\ 0, & \text{otherwise} \end{cases}$$

$$\varepsilon = y_{actual} - y_{prediction}$$

$$w_i^{new} = w_i + \eta \cdot \varepsilon \cdot x_i$$



$\eta = 0.5$ $\theta = 0.5$

n	x_1	x_1	y
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

Note: Perceptron was created with targets (+1, -1). Hence, y = 0 == target = -1

Perceptron Simple Example

1st Epoch

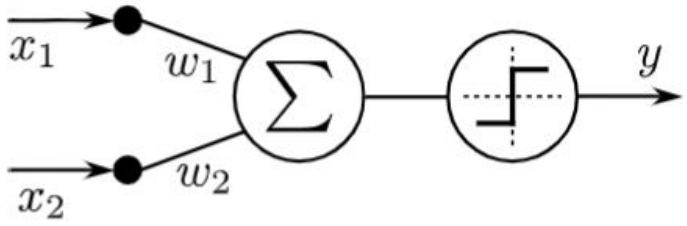
1st Iteration

$$net = 0 \cdot 0.9 + 0 \cdot 0.9 = 0$$

$$\sigma(0) = 0$$

$$\varepsilon = 0 - 0 = 0$$

No error $\therefore$ no update



$\eta = 0.5$ $\theta = 0.5$

n	x_1	x_1	y
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

Perceptron Simple Example

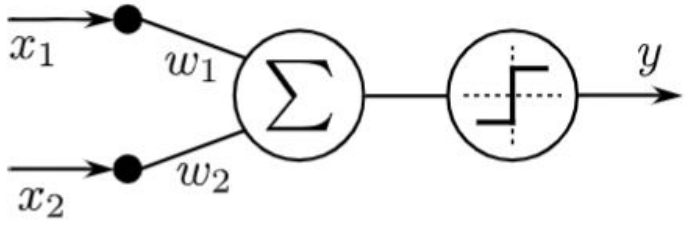
2nd Iteration

$$net = 0 \cdot 0.9 + 1 \cdot 0.9 = 0.9$$
$$\sigma(0.9) = 1$$
$$\varepsilon = 0 - 1 = -1$$

There is an error, hence an update

$$w_1 = 0.9 + 0.5 \cdot (-1) \cdot 0 = 0.9$$
$$w_2 = 0.9 + 0.5 \cdot (-1) \cdot 1 = 0.4$$

These new values are going to go forward to the next iterations.



$\eta = 0.5$ $\theta = 0.5$

n	x_1	x_1	y
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

Perceptron Simple Example

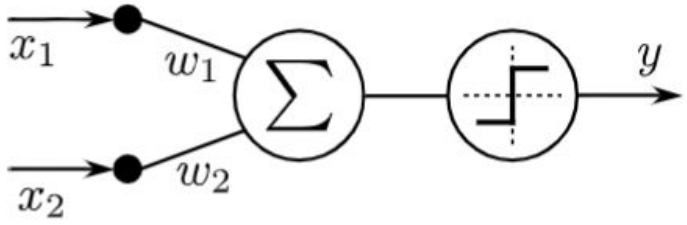
3rd Iteration

$$net = 1 \cdot 0.9 + 0 \cdot 0.4 = 0.9$$
$$\sigma(0.9) = 1$$
$$\varepsilon = 0 - 1 = -1$$

There is an error, hence an update

$$w_1 = 0.9 + 0.5 \cdot (-1) \cdot 1 = 0.4$$
$$w_2 = 0.4 + 0.5 \cdot (-1) \cdot 0 = 0.4$$

These new values are going to go forward to the next iterations.



$\eta = 0.5$ $\theta = 0.5$

n	x_1	x_1	y
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

Perceptron Simple Example

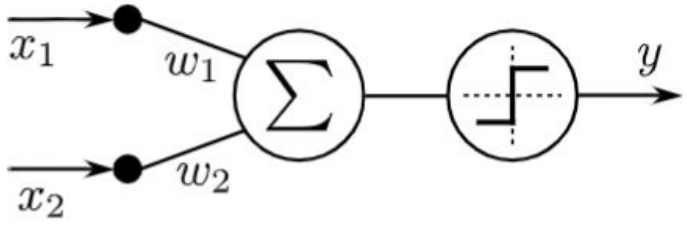
4th Iteration

$$net = 1 \cdot 0.4 + 1 \cdot 0.4 = 0.8$$

$$\sigma(0.8) = 1$$

$$\varepsilon = 1 - 1 = 0$$

No error $\therefore$ no update



$\eta = 0.5$ $\theta = 0.5$

n	x_1	x_1	y
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

ends the **First Epoch**

Perceptron Simple Example

2nd Epoch

1st Iteration

$$net = 0 \cdot 0.4 + 0 \cdot 0.4 = 0$$

$$\sigma(0) = 0$$

$$\varepsilon = 0 - 0 = 0$$

No error $\therefore$ no update

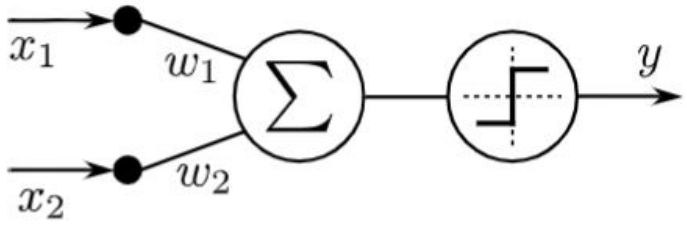
2nd Iteration

$$net = 0 \cdot 0.4 + 1 \cdot 0.4 = 0.4$$

$$\sigma(0.4) = 0$$

$$\varepsilon = 0 - 0 = 0$$

No error $\therefore$ no update



$\eta = 0.5$ $\theta = 0.5$

n	x_1	x_1	y
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

Perceptron Simple Example

3rd Iteration

$$net = 1 \cdot 0.4 + 0 \cdot 0.4 = 0.4$$

$$\sigma(0.4) = 0$$

$$\varepsilon = 0 - 0 = 0$$

No error $\therefore$ no update

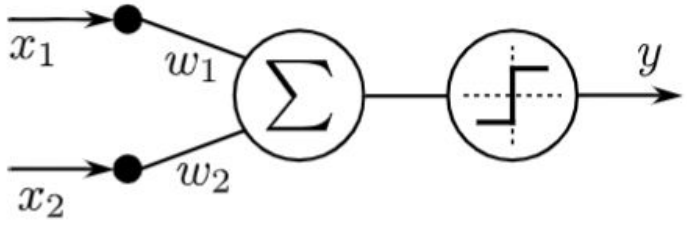
4th Iteration

$$net = 1 \cdot 0.4 + 1 \cdot 0.4 = 0.8$$

$$\sigma(0.8) = 1$$

$$\varepsilon = 1 - 1 = 0$$

No error $\therefore$ no update



$\eta = 0.5$ $\theta = 0.5$

n	x_1	x_1	y
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

There were no updates, hence the training ends with these final values!!!

Perceptron Simple Example - Decision Boundary

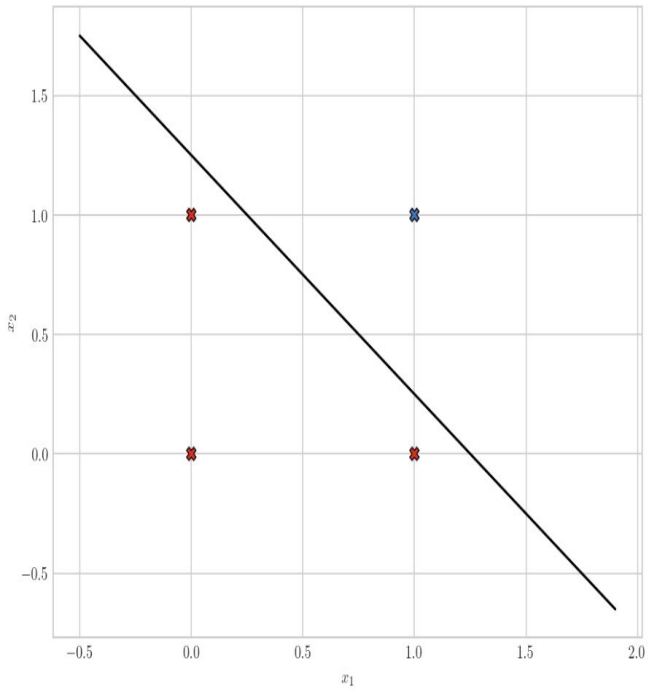
Using the threshold and the net equation the **Decision Boundary** can be calculated as:

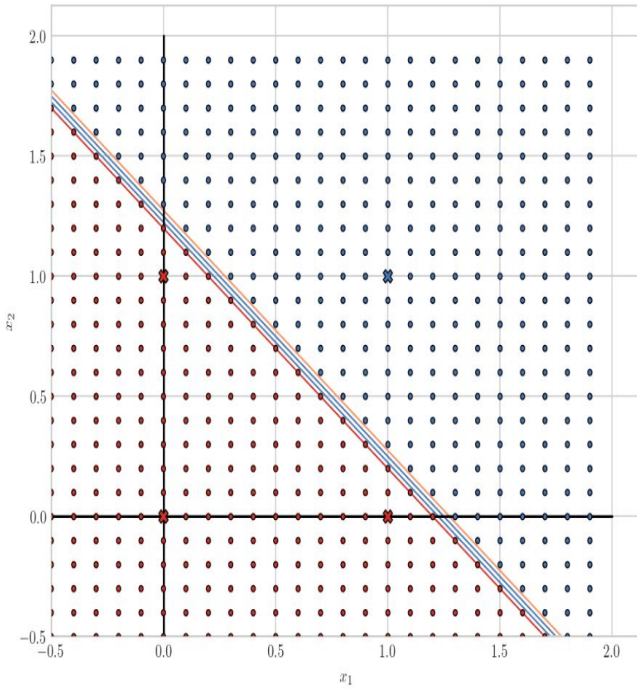
$$\theta = w_1 \cdot x_1 + w_2 \cdot x_2$$

$$x_2 = \frac{\theta}{w_2} - \frac{w_1}{w_2} \cdot x_1$$

with w1= w2 = 0.4
and θ=0.5

$$x_2 = 1.25 - \cdot x_1$$





perceptron_DecisionBoundary_example.ipynb

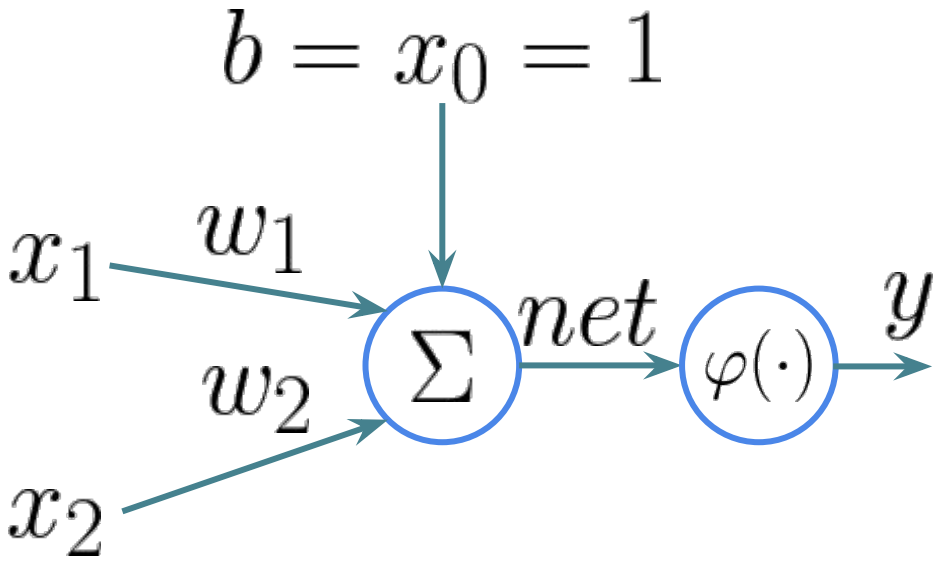
Homework

Repeat the exercise with the Logistic OR, NOR and XOR.

Do only 3 **Epochs** (4 iterations each epoch) and see if the algorithm converges.

Bias Trick

Adding the bias would help, yet, the bias has to be treated differently trainable



$$net = w_0 * x_0 + w_1 * x_1 + w_2 * x_2$$

$$y = \varphi(net)$$

This way it can be trained similarly to the other weights

$$\varphi = \begin{cases} +1 & \text{if} \quad net \geq 0 \\ -1 & \text{if} \quad net < 0 \end{cases}$$

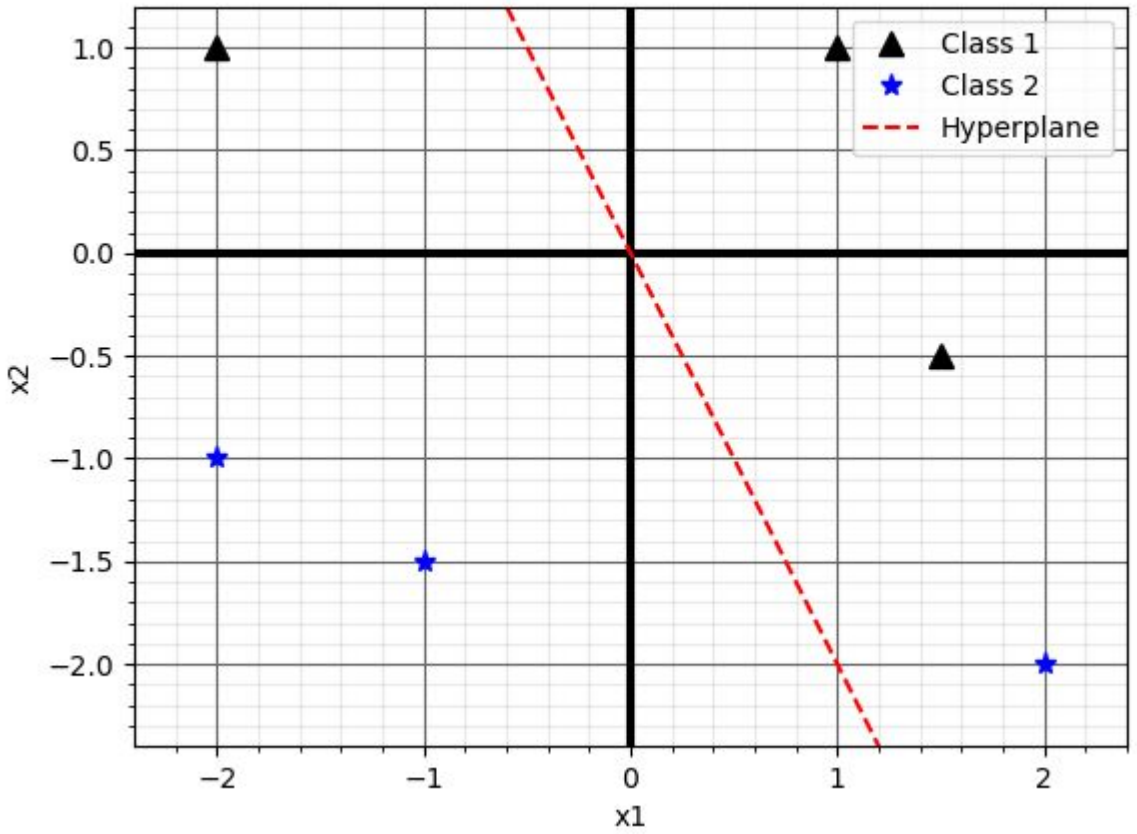
Example

$$\alpha = 0.2 \qquad w = \begin{pmatrix} 0 \\ 1 \\ 0.5 \end{pmatrix}$$

$$x = \begin{bmatrix} 1, & 1 \\ 2, & -2 \\ -1, & -1.5 \\ -2, & -1 \\ -2, & 1 \\ 1.5, & -0.5 \end{bmatrix} \quad t = \begin{bmatrix} +1 \\ -1 \\ -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$

At $y = 0$ the resulting hyperplane is:

$$0 = w_0 \cdot 1 + w_1 \cdot x_1 + w_2 \cdot x_2$$
$$0 = 0 + x_1 + 0.5x_2$$
$$\Rightarrow x_2 = 2x_1$$



Example

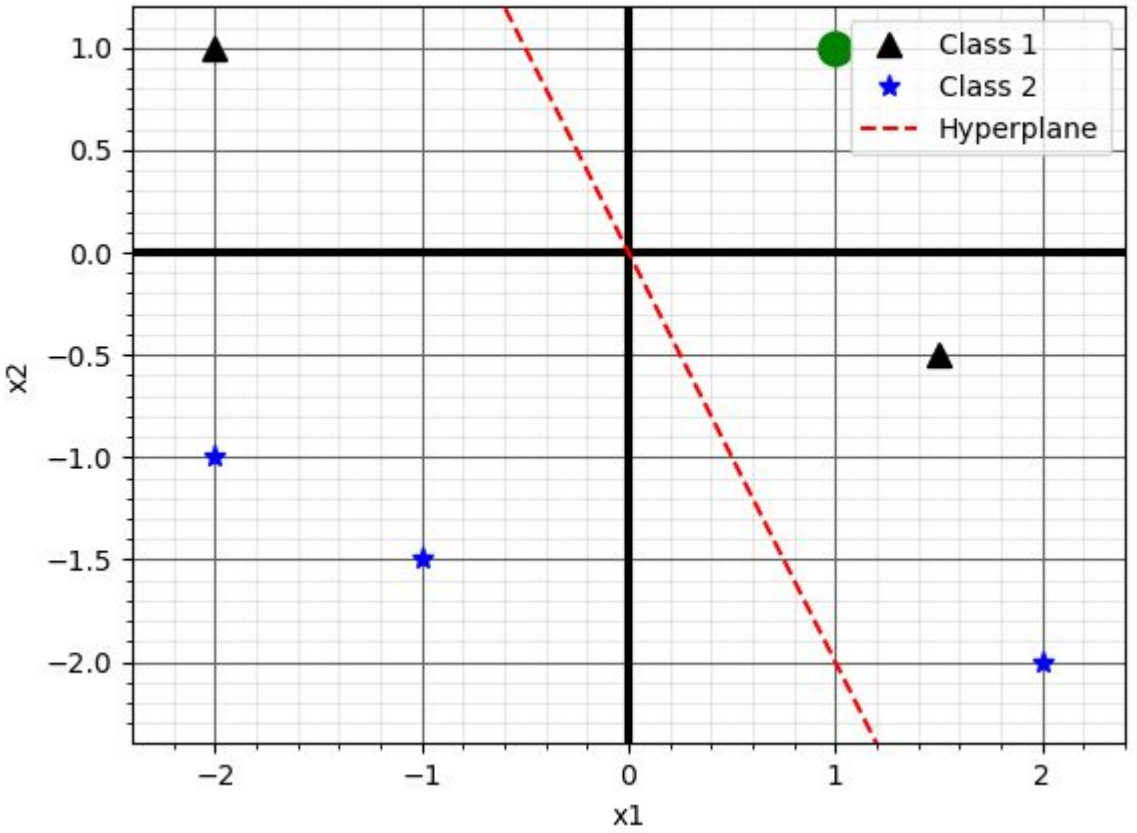
$$\alpha = 0.2 \qquad w = \begin{pmatrix} 0 \\ 1 \\ 0.5 \end{pmatrix}$$

$$x_1 = 1,\ x_2 = 1$$

$$w^T x > 0$$

Correct classification, **No** action

$$x = \begin{bmatrix} 1, & 1 \\ 2, & -2 \\ -1, & -1.5 \\ -2, & -1 \\ -2, & 1 \\ 1.5, & -0.5 \end{bmatrix} t = \begin{bmatrix} +1 \\ -1 \\ -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$



Example

$$x = \begin{bmatrix} 1, & 1 \\ 2, & -2 \\ -1, & -1.5 \\ -2, & -1 \\ -2, & 1 \\ 1.5, & -0.5 \end{bmatrix} t = \begin{bmatrix} +1 \\ -1 \\ -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$

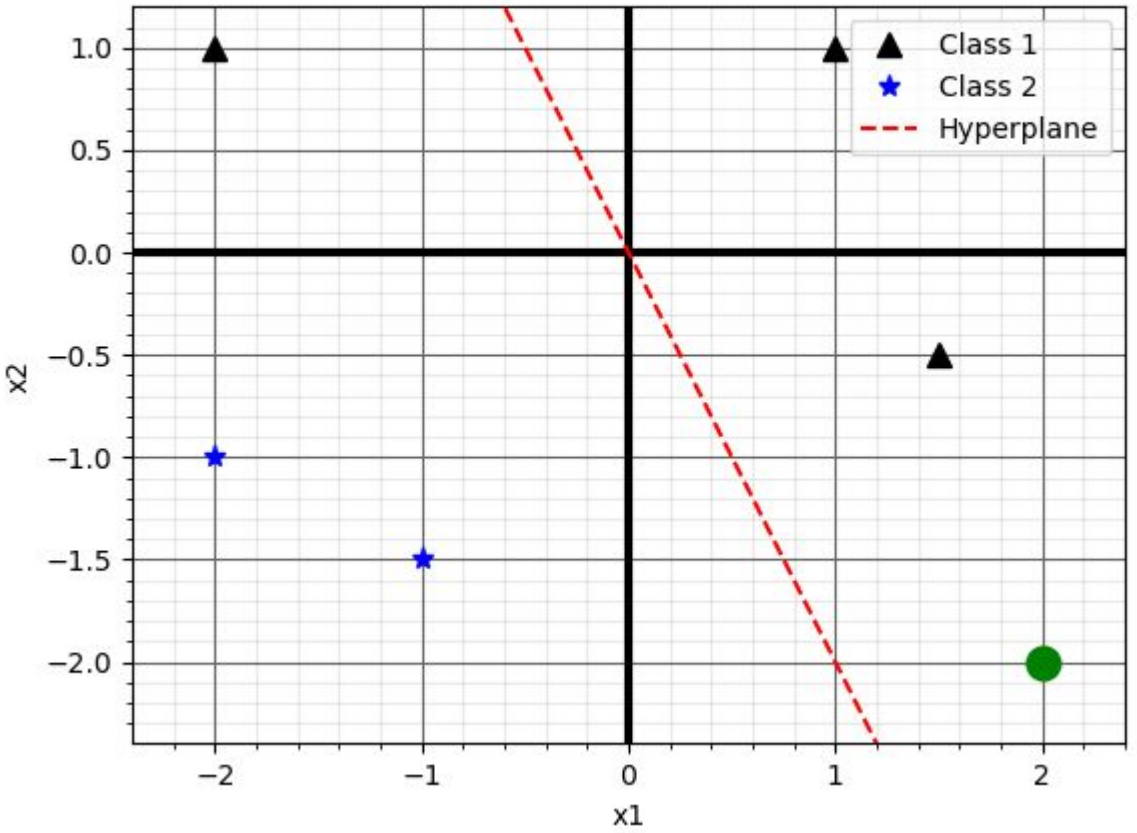
$$\alpha = 0.2 \qquad w = \begin{pmatrix} 0 \\ 1 \\ 0.5 \end{pmatrix}$$

$$x_1 = 2,\ x_2 = -2$$

$$w^T x > 0$$

Incorrect classification,

$$w_0 = w_0 - 0.2 * (1)$$
$$w_1 = w_1 - 0.2 * (2)$$
$$w_2 = w_2 - 0.2 * (-2)$$



Example

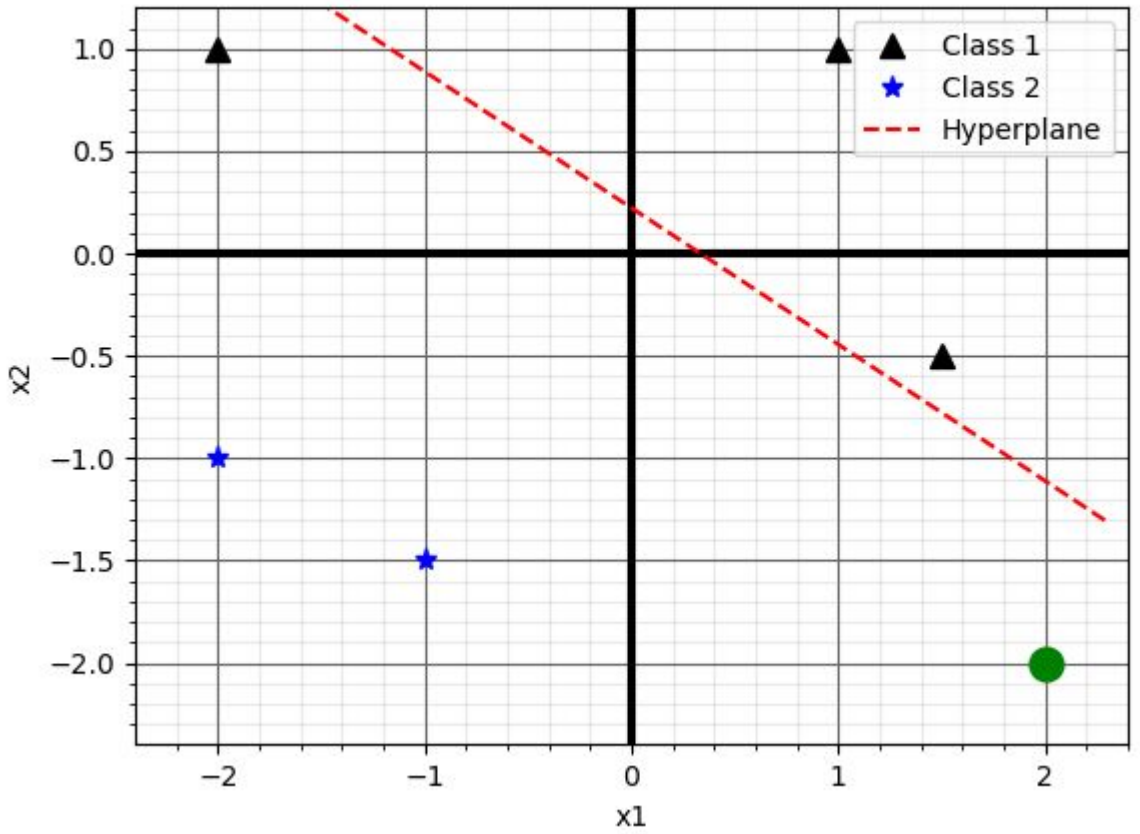
$$\alpha = 0.2 \qquad w = \begin{pmatrix} -0.2 \\ 0.6 \\ 0.9 \end{pmatrix}$$

$$x_1 = 2,\ x_2 = -2$$

$$w^T x < 0$$

Correct classification, **No** action

$$x = \begin{bmatrix} 1, & 1 \\ 2, & -2 \\ -1, & -1.5 \\ -2, & -1 \\ -2, & 1 \\ 1.5, & -0.5 \end{bmatrix} t = \begin{bmatrix} +1 \\ -1 \\ -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$



Example

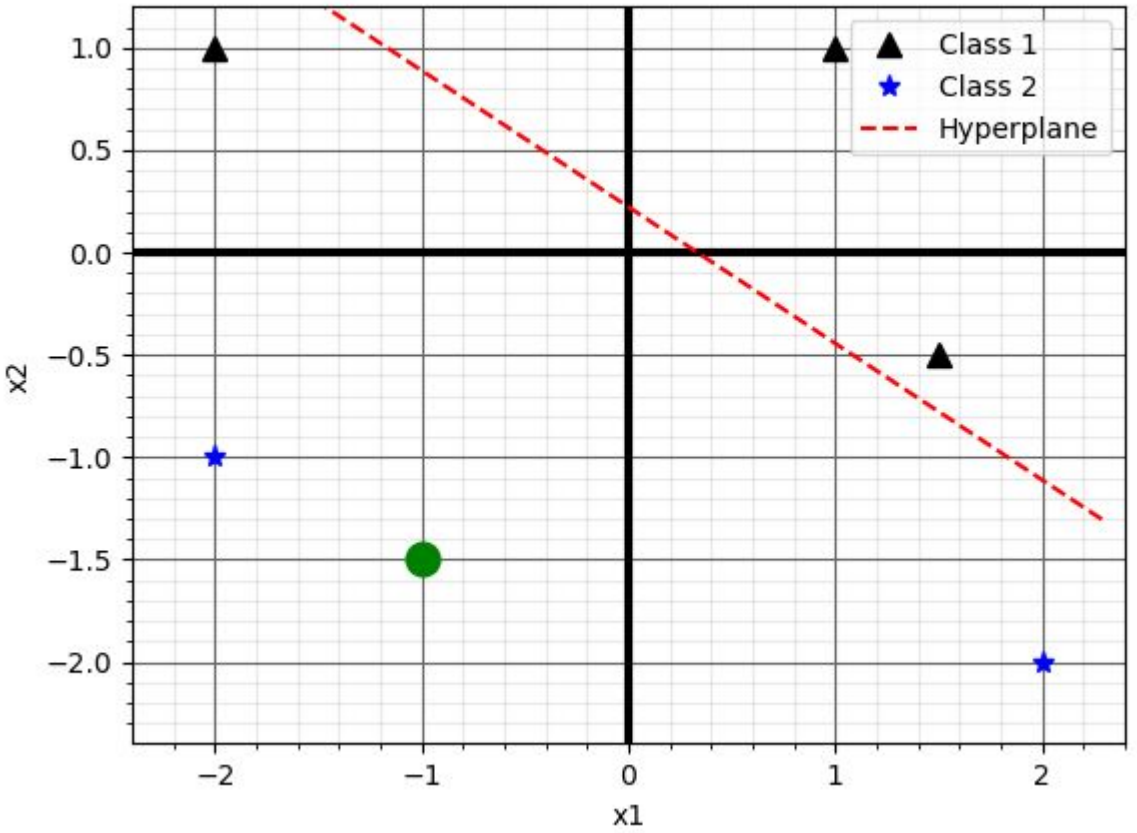
$$\alpha = 0.2 \qquad w = \begin{pmatrix} -0.2 \\ 0.6 \\ 0.9 \end{pmatrix}$$

$$x = \begin{bmatrix} 1, & 1 \\ 2, & -2 \\ -1, & -1.5 \\ -2, & -1 \\ -2, & 1 \\ 1.5, & -0.5 \end{bmatrix} t = \begin{bmatrix} +1 \\ -1 \\ -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$

$$x_1 = -1, \; x_2 = -1.5$$

$$w^T x < 0$$

Correct classification, **No** action



Example

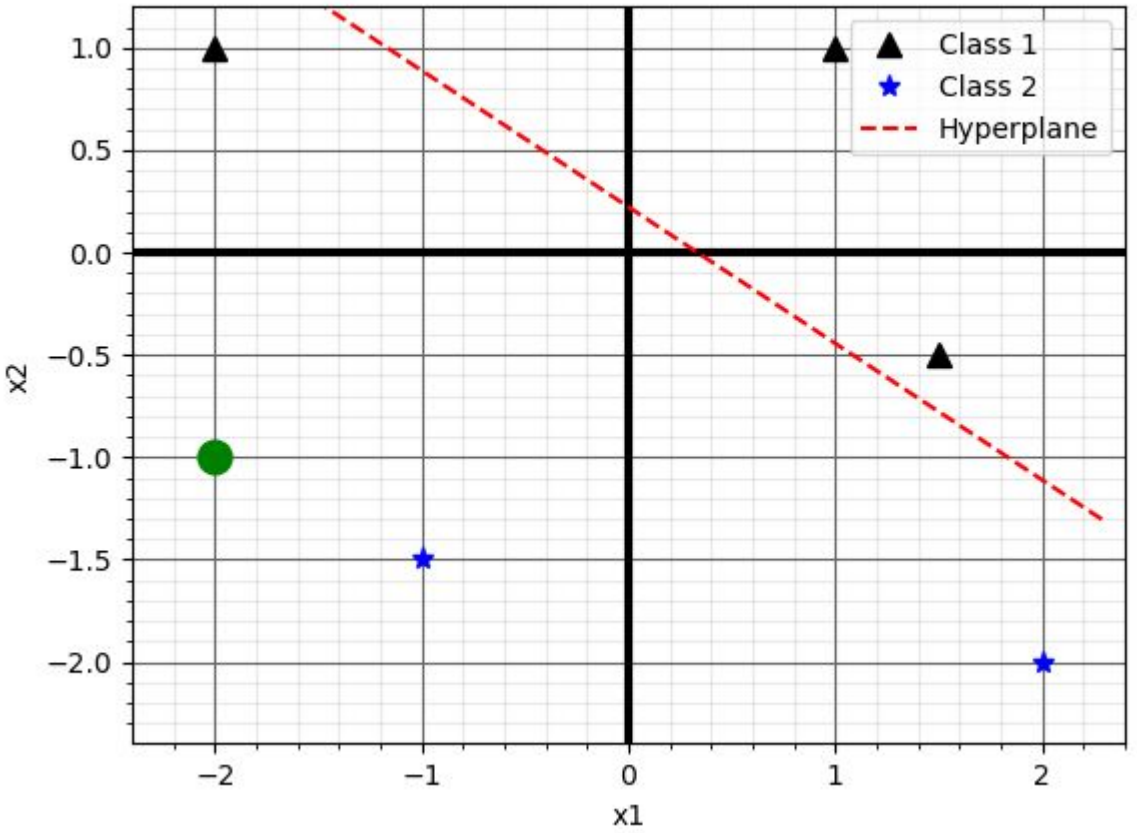
$$\alpha = 0.2 \qquad w = \begin{pmatrix} -0.2 \\ 0.6 \\ 0.9 \end{pmatrix}$$

$$x_1 = -2, \;\; x_2 = -1$$

$$w^T x < 0$$

Correct classification, **No** action

$$x = \begin{bmatrix} 1, & 1 \\ 2, & -2 \\ -1, & -1.5 \\ -2, & -1 \\ -2, & 1 \\ 1.5, & -0.5 \end{bmatrix} \quad t = \begin{bmatrix} +1 \\ -1 \\ -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$



Example

$$\alpha = 0.2 \qquad w = \begin{pmatrix} -0.2 \\ 0.6 \\ 0.9 \end{pmatrix}$$

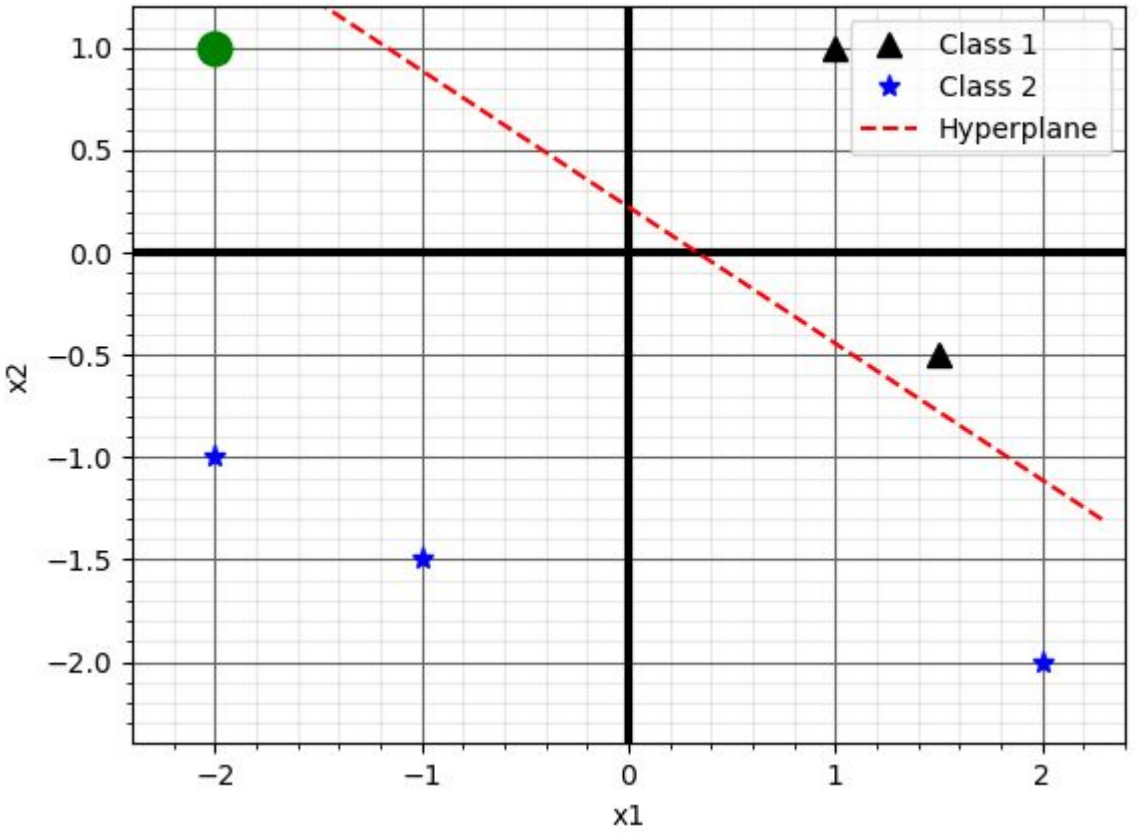
$$x = \begin{bmatrix} 1, & 1 \\ 2, & -2 \\ -1, & -1.5 \\ -2, & -1 \\ -2, & 1 \\ 1.5, & -0.5 \end{bmatrix} t = \begin{bmatrix} +1 \\ -1 \\ -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$

$$x_1 = -2, \; x_2 = 1$$

$$w^T x < 0$$

Incorrect classification,

$$w_0 = w_0 + 0.2 * (1)$$
$$w_1 = w_1 + 0.2 * (-2)$$
$$w_2 = w_2 + 0.2 * (1)$$



Example

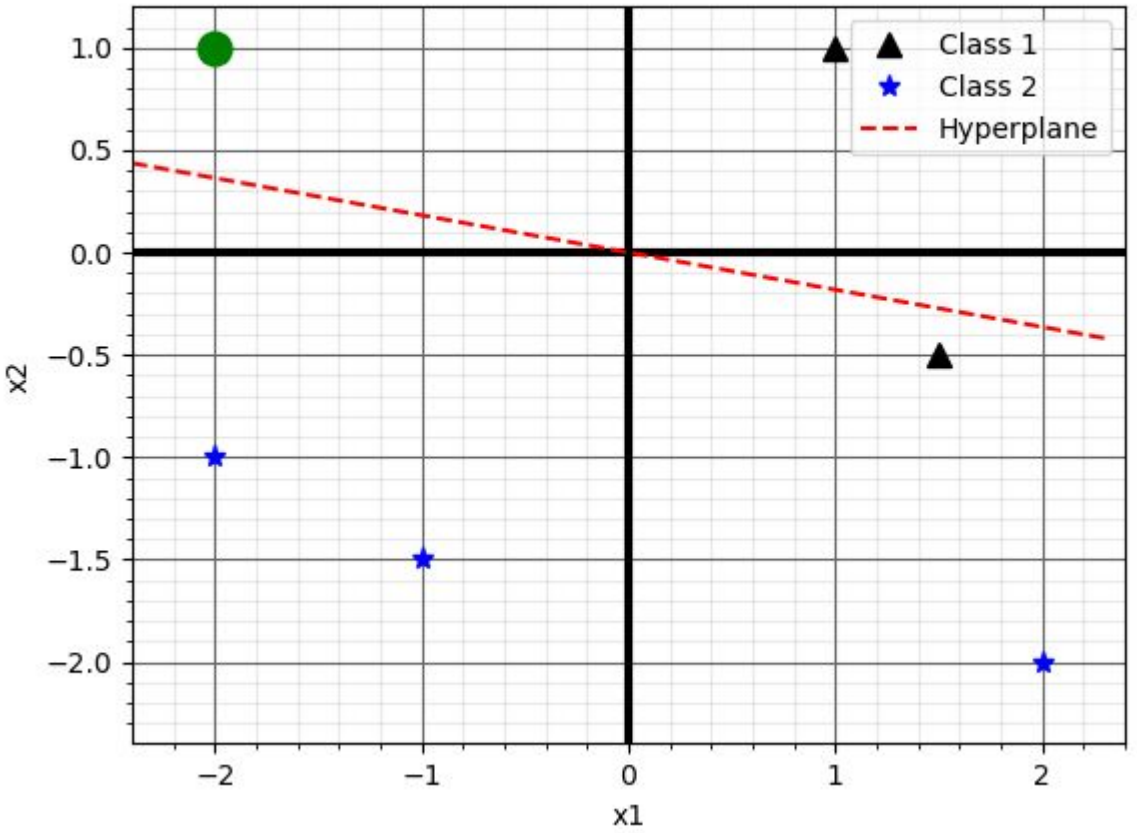
$$\alpha = 0.2 \qquad w = \begin{pmatrix} 0 \\ 0.2 \\ 1.1 \end{pmatrix}$$

$$x_1 = -2, \; x_2 = 1$$

$$w^T x > 0$$

Correct classification,

$$x = \begin{bmatrix} 1, & 1 \\ 2, & -2 \\ -1, & -1.5 \\ -2, & -1 \\ -2, & 1 \\ 1.5, & -0.5 \end{bmatrix} t = \begin{bmatrix} +1 \\ -1 \\ -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$



Example

$$\alpha = 0.2 \qquad w = \begin{pmatrix} 0 \\ 0.2 \\ 1.1 \end{pmatrix}$$

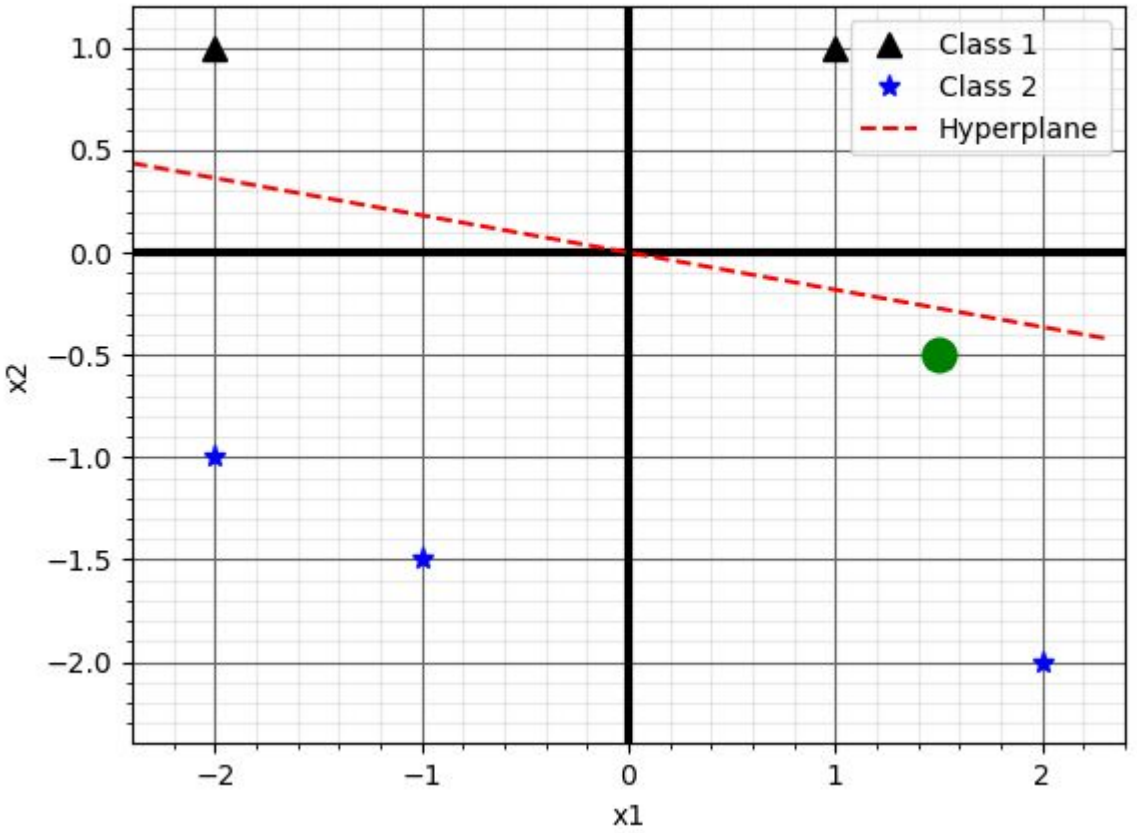
$$x_1 = 1.5, \; x_2 = -0.5$$

$$w^T x < 0$$

Incorrect classification,

$$w_0 = w_0 + 0.2 * (1)$$
$$w_1 = w_1 + 0.2 * (1.5)$$
$$w_2 = w_2 + 0.2 * (-0.5)$$

$$x = \begin{bmatrix} 1, & 1 \\ 2, & -2 \\ -1, & -1.5 \\ -2, & -1 \\ -2, & 1 \\ 1.5, & -0.5 \end{bmatrix} t = \begin{bmatrix} +1 \\ -1 \\ -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$



Example

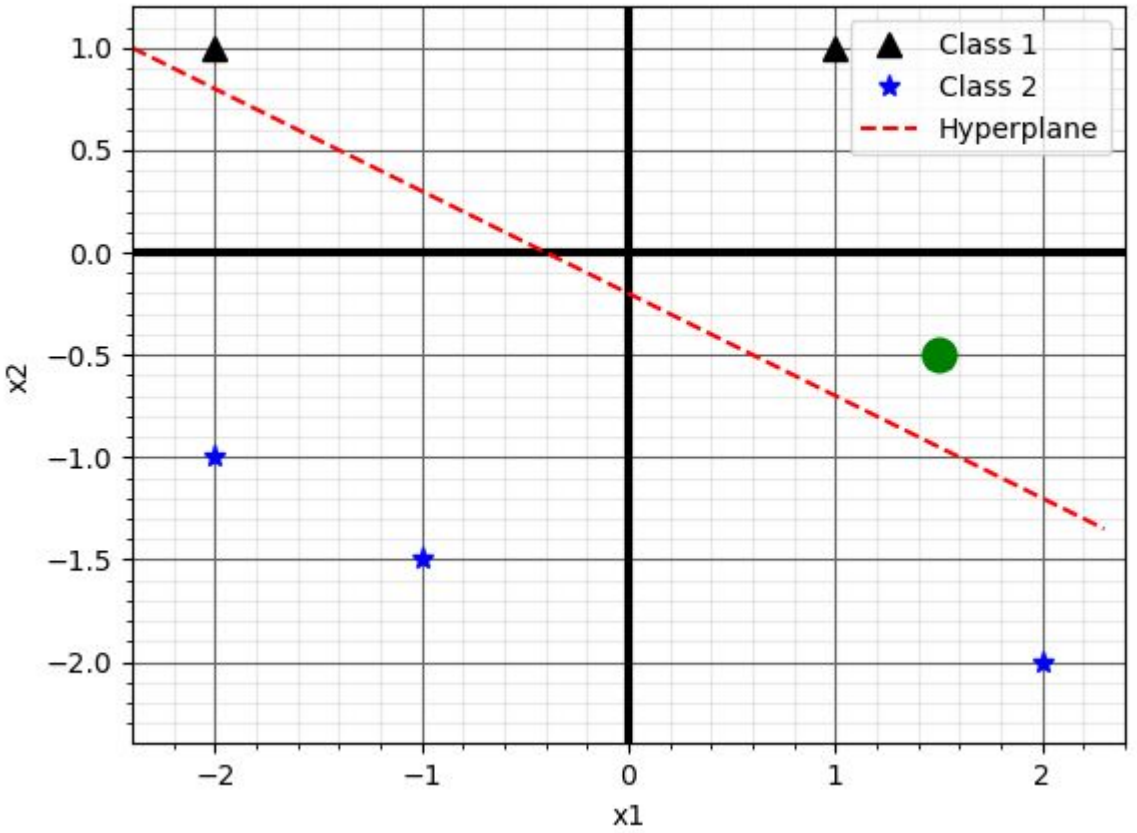
$$\alpha = 0.2 \qquad w = \begin{pmatrix} 0.2 \\ 0.5 \\ 1 \end{pmatrix}$$

$$x_1 = 1.5, \; x_2 = -0.5$$

$$w^T x > 0$$

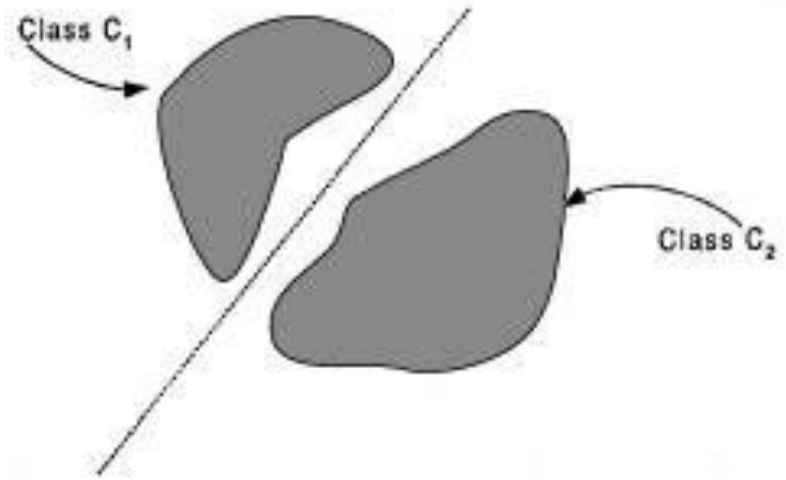
Correct classification,

$$x = \begin{bmatrix} 1, & 1 \\ 2, & -2 \\ -1, & -1.5 \\ -2, & -1 \\ -2, & 1 \\ 1.5, & -0.5 \end{bmatrix} t = \begin{bmatrix} +1 \\ -1 \\ -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$



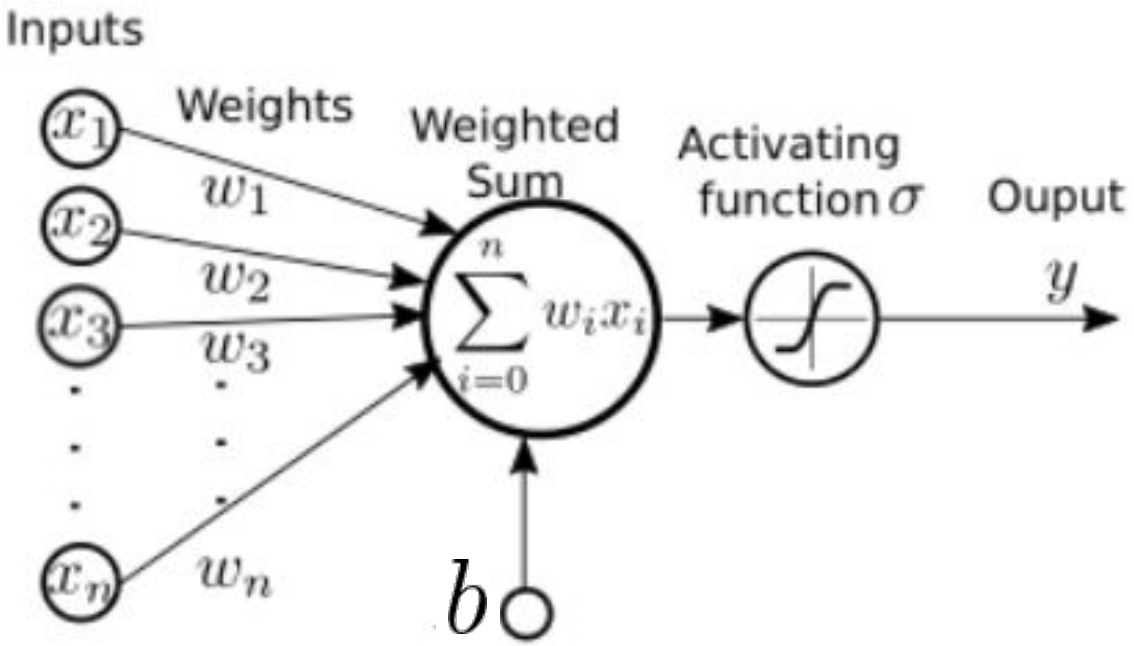
Perceptron Convergence Theorem

- The theorem states that for any data set which is **linearly separable**, the perceptron learning rule is **guaranteed** to find a solution in a **finite** number of iterations.



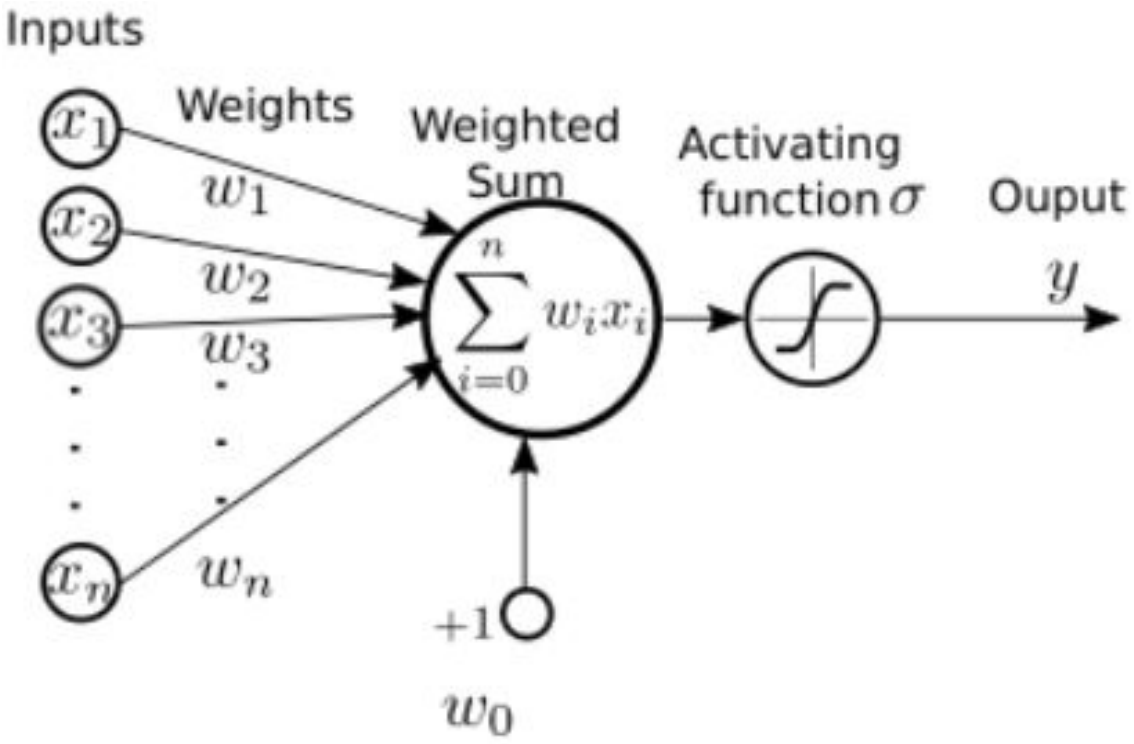
Perceptron with bias treated separately

- Algorithm Perceptron Pseudo-code
  Input network:
  - Training set S = $\{(x_1, t_1), (x_2, t_3), \ldots, (x_s, t_s)\}$
  - Learning rate α ((0 − 1])
  - Maximum number of iterations maxIter
- Init network:
  - Initialize the weights to a small random value
- Train network:
  - Loop until w new ! = w old && iter < maxIter):
    - Select random input pattern (x_s, t_s):
    - Compute network activation $net_s = w_i \times x_{si} + b)$
    - Compute output $y_s = \sigma(net_s)$
    - if y_s ! = t_s
      - $w_{new} = w_{old} + \alpha(t_s - y_s)x_i$
      - $b_{new} = b_{old} + \alpha(t_s - y_s)x_i$
    - else
    - $w_{new} = w_{old}$
- Output network: Return weights

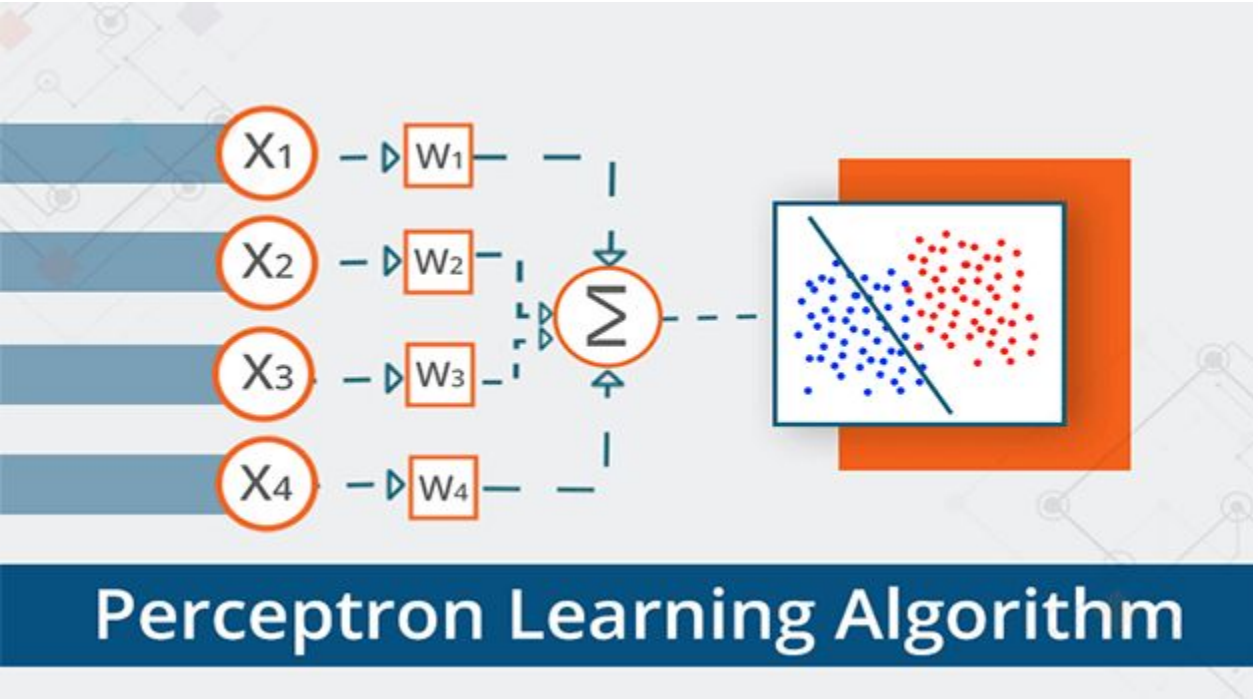


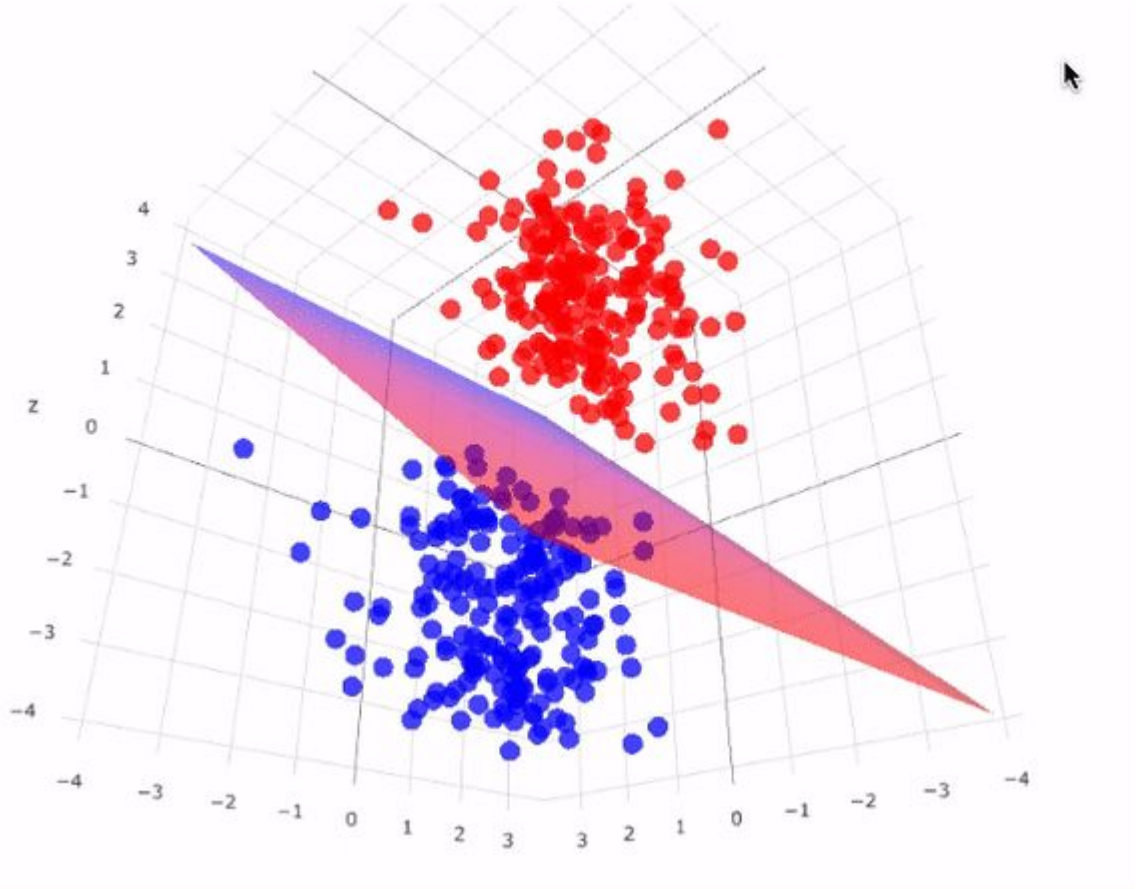
Perceptron

- Algorithm Perceptron Pseudo-code
  Input network:
  - Training set S = {(x_1 ,t_1),(x_2 ,t_3),...,(x_s ,t_s)}
  - Learning rate α ((0 − 1])
  - Maximum number of iterations maxIter
- Init network:
  - Initialize the weights to a small random value
- Train network:
  - Loop until w new ! = w old && iter < maxIter):
    - Select random input pattern (x_s ,t_s):
    - Compute network activation net_s =Σw_i × x_{si})
    - Compute output y_s = σ(net_s)
    - if y_s ! = t_s
      - w_{new}= w_{old}+ α(t_s − y_s)x_i
    - else
    - w_{new}= w_{old}
- Output network: Return weights



Perceptron 2D & 3D





https://colab.research.google.com/github/mohammedterry/ANNs/blob/master/ML_ANN.ipynb