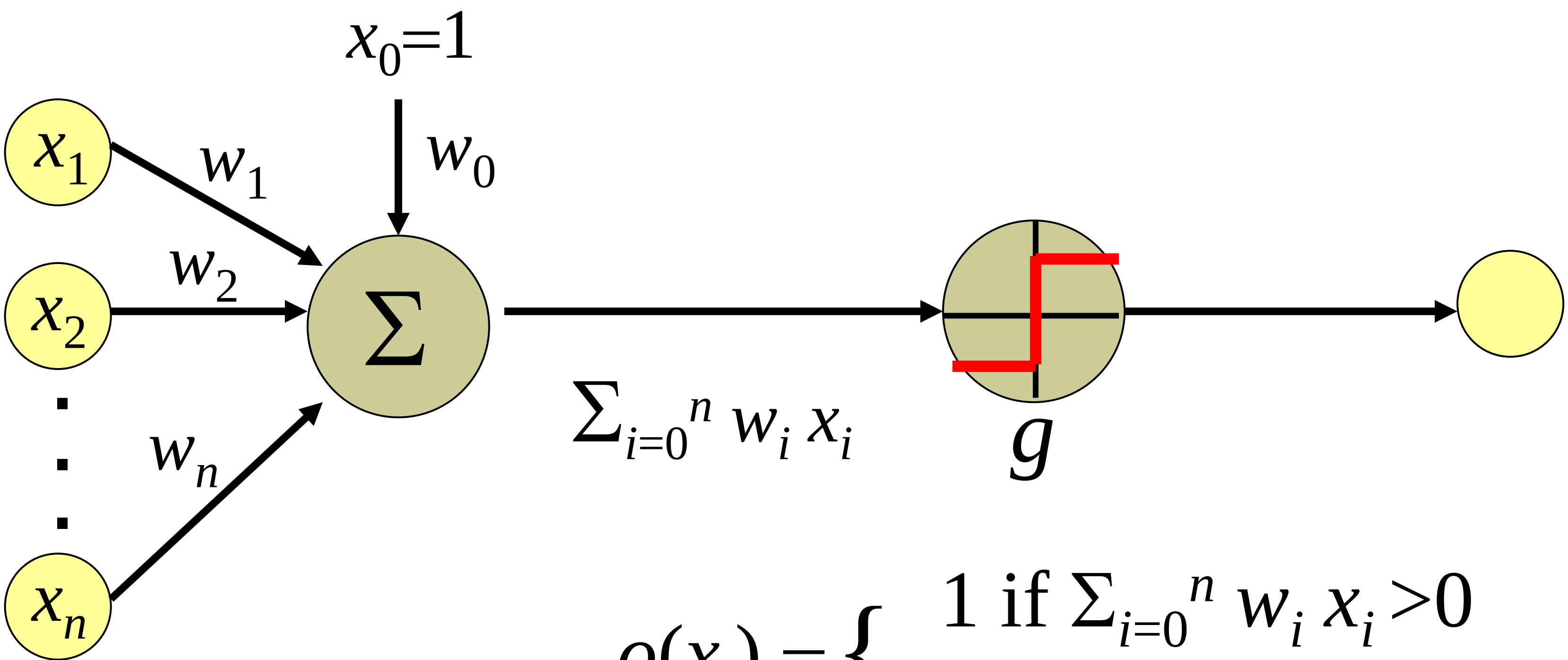
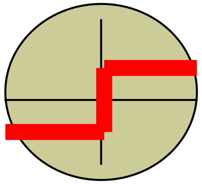
Perceptron

- Linear threshold unit (LTU)



$$o(x_i) = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i\, x_i > 0 \\ -1 \text{ otherwise} \end{cases}$$

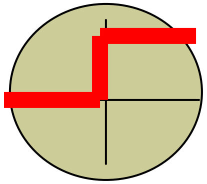
Possibilities for function *g*



Sign function

sign(x) = +1, if $x > 0$

−1, if $x \le 0$

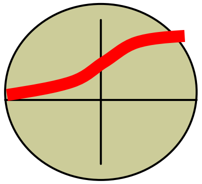


Step function

step(x) = 1, if $x >$ threshold

0, if $x \le$ threshold

(in picture above, threshold = 0)

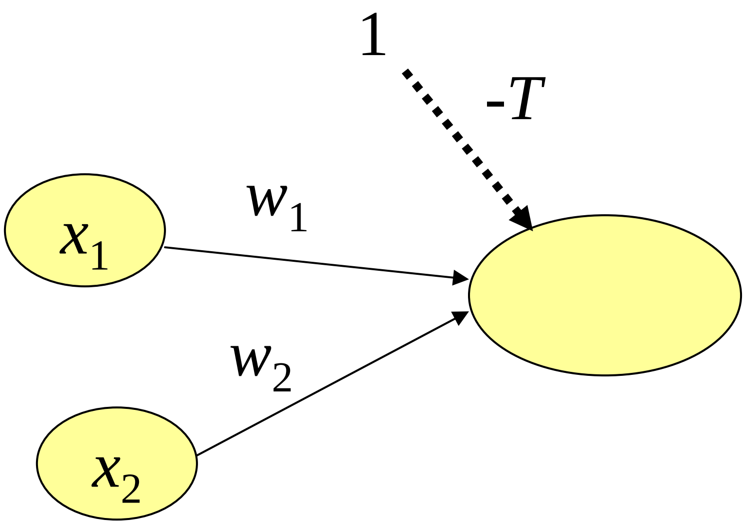


Sigmoid (logistic) function

sigmoid(x) = $1/(1+e^{-x})$

Adding an extra input with activation $x_0 = 1$ and weight $w_{i,0} = -T$ (called the ***bias weight***) is equivalent to having a threshold at T. This way we can always assume a 0 threshold.

Using a Bias Weight to Standardize the Threshold



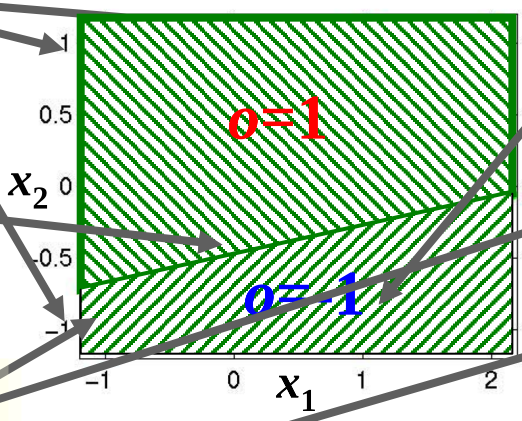
$$w_1x_1 + w_2x_2 < \boldsymbol{T}$$

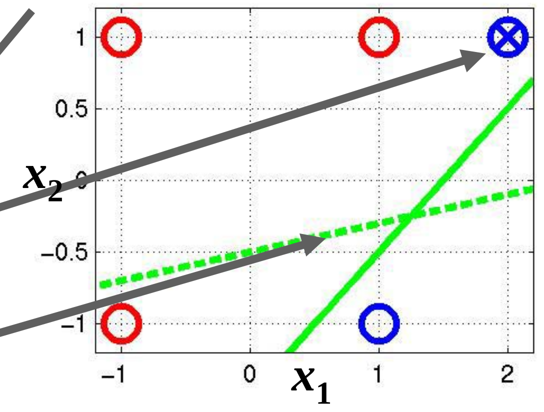
$$w_1x_1 + w_2x_2 - T < \mathbf{0}$$

Perceptron Learning Rule

$t = 1$

$t = -1$





$w = [0.25 \ –0.1 \ 0.5]$

$x_2 = 0.2\,x_1 – 0.5$

$(x, t)=([2,1], -1)$

$(x, t)=([-1,-1], 1)$

$o = \text{sgn}(0.45-0.6+0.3)$

$=1$

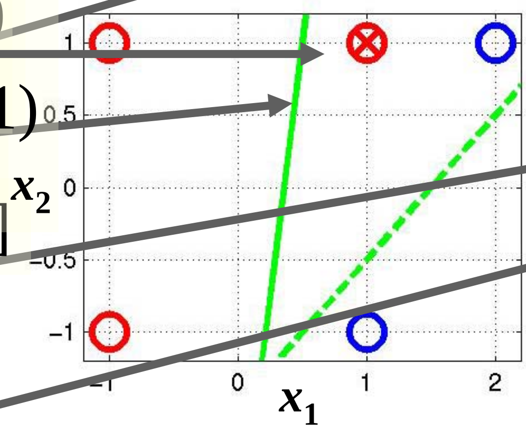
$(x, t)=([1,1], 1)$

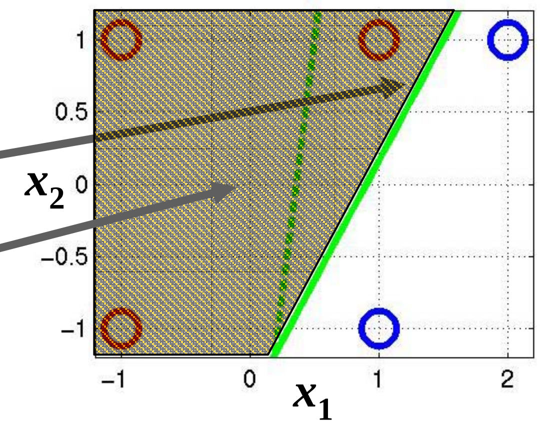
$o = \text{sgn}(0.25–0.7+0.1)$

$= –1$

$\Delta w = [0.2 \ –0.2 \ –0.2]$

$\Delta w = [–0.2 \ –0.4 \ –0.2]$





$\Delta w = [0.2 \ 0.2 \ 0.2]$

$-0.5x_1+0.3x_2+0.45>0 \Rightarrow o = 1$

The Perceptron Algorithm
Rosenblatt, 1956

Given a linearly separable training set S and learning rate $\eta > 0$ and the initial weight vector, bias: $w^0 = \mathbf{0}, \ b_0 = 0$ and let

$$R = \max_{1 \leqslant i \leqslant \ell} ||x^i||, \ \ k = 0.$$

The Perceptron Algorithm

(Primal Form)

$$R = \max_{1 \leqslant i \leqslant \ell} ||x^i||, \quad k = 0.$$

Repeat: $for \;\; i = 1 \;\; to \;\; \ell$

$$if \;\; y_i(\langle w^k \cdot x^i \rangle + b_k) \leqslant 0 \;\; then$$

$$w^{k+1} \leftarrow w^k + \eta y_i x^i$$

$$b_{k+1} \leftarrow b_k + \eta y_i R^2$$

$$k \leftarrow k + 1$$

$$end \;\; if$$

$$end \;\; for$$

until no mistakes made within the for loop return:

$k, \;\; (w^k, b_k)$. What is k ?

$$y_i\left(\left\langle w^{k+1} \cdot x^i \right\rangle + b_{k+1}\right) > y_i\left(\left\langle w^k \cdot x^i \right\rangle + b_k\right) \ ?$$

$$w^{k+1} \leftarrow w^k + \eta y_i x^i \ \ and \quad b_{k+1} \leftarrow b_k + \eta y_i R^2$$

$$y_i\left(\left\langle w^{k+1} \cdot x^i \right\rangle + b_{k+1}\right)$$

$$= y_i\left(\left\langle (w^k + \eta y_i x^i) \cdot x^i \right\rangle + b_k + \eta y_i R^2\right)$$

$$= y_i\left(\left\langle w^k \cdot x^i \right\rangle + b_k\right) + y_i\left(\eta y_i \left\langle x^i \cdot x^i \right\rangle + R^2\right)$$

$$= y_i\left(\left\langle w^k \cdot x^i \right\rangle + b_k\right) + \eta\left(\left\langle x^i \cdot x^i \right\rangle + R^2\right)$$

The Perceptron Algorithm

(STOP in Finite Steps)

Theorem (Novikoff)

Let S be a non-trivial training set, and let $R = \max_{1 \leqslant i \leqslant \ell} ||x^i||_2$.

Suppose that there exists a vector $w_{opt} \in R^n, ||w_{opt}|| = 1$

and $y_i(\langle w_{opt} \cdot x^i \rangle + b_{opt}) \geqslant \gamma, \forall\ 1 \leqslant i \leqslant \ell$. Then the number

of mistakes made by the on-line perceptron algorithm

on S is at most $(\frac{2R}{\gamma})^2$.

The Perceptron Algorithm

(Dual Form) $w = \sum_{i=1}^{l} \alpha_i y_i x^i$

Given a linearly separable training set S and

$$\alpha = \mathbf{0}, \ \ \alpha \in R^l, b = 0, R = \max_{1 \leqslant i \leqslant l} ||x_i||$$

Repeat:

$$for \ \ i = 1 \ \ to \ \ l$$

$$if \ \ y_i(\sum_{j=1}^{l} \alpha_j y_j \langle x^j \cdot x^i \rangle + b) \leqslant 0 \ \ then$$

$$\alpha_i \leftarrow \alpha_i + 1; \ \ \ b \leftarrow b + y_i R^2$$

$$end \ \ if$$

$$end \ \ for$$

until no mistakes made within the for loop return: (α, b)

What We Got in the Dual Form Perceptron Algorithm?

- The number of updates equals: $\sum_{i=1}^{l} \alpha_i = ||\alpha||_1 \leqslant (\frac{2R}{\gamma})^2$

- $\alpha_i > 0$ implies that the training point (x^i, y_i) has been misclassified in the training process at least once.

- $\alpha_i = 0$ implies that removing the training point (x^i, y_i) will not affect the final results

- The training data only appear in the algorithm through the entries of the Gram matrix, $G \in R^{l \times l}$ which is defined below:

$$G_{ij} = \langle x^i, x^j \rangle$$